AB

例題方式

Quick BASIC プログラミングテキスト

▼小島政行著

ST

PC 9800シリーズ/AX対応

例題方式

Quick BASIC プログラミングテキスト

▶小島政行著

サイエンテック

カバーデザイン

Kingyo COMPANY

PC 9800シリーズ/AX対応

例題方式

Quick BASIC プログラミングテキスト

▶小島政行著

サイエンテック

本書の目的と構成

目的　本書は、これまでプログラムを組んだことのない人を対象に、Quick BASICを用いてプログラムの基本的な要素や記述の仕方などをやさしく解説し、「プログラミングの第1歩」を踏み出してもらうために書かれています。

構成　**■第1部　基本操作**

第1部では、プログラムを開発する道具としてQuick BASICの操作方法を解説します。Quick BASICの立ち上げと終了、プログラムの作成と実行、ファイルへの保存など、一連の手順を解説します。

■第2部　プログラミングⅠ(初級編)

第2部では、プログラミング言語としてのQuick BASICを解説します。
はじめてのプログラミング言語としてQuick BASICを学ぶ人を対象に、プログラムの基本的な要素、組み立て方を例題を中心に学習します。

■第3部　プログラミングⅡ(中級編)

第3部では、ファイルとの入出力やグラフィックといった、少し難解な課題について解説します。

テキストの選択・本書の特徴

Quick BASICは、従来のBASICとはプログラミング作法がまったく異なります。行番号を使ってプログラム例の解説をしているテキストを用いてQuick BASICを学習するならば、百害あって一利無しです。

Quick BASICの学習には、Quick BASICを題材としたテキストを使ってください。Quick BASICの本でもGOTO文を解説している本は、読まない方がよいでしょう。

本書は、Quick BASICの特徴を生かし、構造化プログラミングを学習するための本です。

学習の進め方

本書は、例題方式を採用しています。各テーマ(セクション)には、そのはじめに主題についての説明があります。これを見て、例題を解いてください。[解説]には、その例題に付随する事項を、項目を分けて解説してあります。

例題をひとつずつ解いて、Quick BASICによる構造化プログラミングをマスターしてください。

謝意

本書作成にあたっては、マイクロソフト株式会社より「Microsoft Quick BASIC」一式を提供いただきました。

また、キヤノン販売株式会社より、32ビットAXパソコン「AXi DX-20」(一式)を借用いたしました。マイクロソフト株式会社ならびにキヤノン販売株式会社に、厚く感謝の意を表します。

また、本書作成にあたっては、株式会社サイエンテックの全社的な協力を得ました。なお、例題の実行確認ならびに校正を担当した大西理恵さんに感謝いたします。

お願い

(1)本書の内容に関するお問い合わせは、下記宛にFAXにてお願いいたします。

㈲アプライド ナレッジ 編集部

電話にてFAX送付の旨を告げた上で、お問い合わせ状をFAXしてください。

TEL 045－311－3795　　FAX 045－311－5883

(2)本書で使用しましたAXパソコン キヤノンAXi DX-20に関するお問い合わせは、

キヤノン販売㈱ AX販売企画部　TEL 03－455－9127

までお願いいたします。

目　　次

第2章　コンソールとの入出力　55

第3章　分岐と繰り返し　71

第5章　データ

第6章　関数　121

第3部　プログラミングII（中級編）　133

第1章　プロシージャ　135

第2章　ファイル 147

第3章　グラフィックⅡ　171

第 1 部

基 本 操 作

第1章　準　備

第1章では、Quick　BASICによるプログラミングの学習をはじめる前の準備として、Quick　BASICのインストールや立ち上げ、終了などを行ないます。

第1章の概要

■Quick　BASICの特徴

Quick　BASICは、いままでのBASICの概念を打ち破った新しい構造化プログラミング言語です。

■使用機器

Quick　BASICは、PC-9801シリーズとAXパソコンの両方に対応しています。
この本では、以下の2機種でその動作を確認しています。

●PC-9801シリーズ　　PC-9801　VX21(日本電気)

●AXパソコン　　AXi　DX-20(キヤノン)

■キーの表記

PC-9801シリーズとAXパソコンでは、キートップ(各キーの表示)が違います。この本では、わずかな違いのものについては、PC-9801シリーズのキートップを、大きく違うものについては両方を併記しています。

■システムの構成

Quick　BASICは、QB.EXE、BC.EXEなどのファイルから構成される、総合的なプログラミング開発環境を提供します。

■バックアップ

購入した大切なQuick　BASICは、バックアップをとっておきましょう。

■インストール

Quick　BASICには、インストールのためのユーティリティが付いています。簡単にインストールできます。

■日本語の使用

■パスの設定

ルートディレクトリからQuick　BASICを使用するためには、パスの設定が必要です。

■立ち上げ

■終了

Quick BASICの特徴

BASIC

BASICというプログラミング言語は、その名の由来(Beginner's All Purpose Symbolic Instruction Code)が示すように、初心者向けの多目的なプログラミング言語です。パソコンの出始めの頃には、BASIC以外の確かな言語がなかったために、市販されるかなり大きなソフトウェア(プログラム)がBASICで書かれていました。しかし、BASICに内在する基本的な欠陥のために、大きなプログラムを効率よく書くことは至難のわざでした。その後、より速く動くより大きなプログラムの開発に適したC言語のパソコン版が出現し、多くのプログラムがC言語で書かれるようになりました。C言語の出現により、BASICは初学者の「お遊びのプログラミング言語」になりさがってしまいました。

しかし、その内在する欠陥をすべて取り除き、機能アップした「Quick BASIC」の登場により、BASICは再びプロの目にとまる「プロ好みのプログラミング言語」になりました。

Quick BASICの特徴

Quick BASICは、以下の特徴を持ったまったく新しいBASICです。

■Quick BASICの特徴

(1)構造化のためのステートメント

SELECT CASE, DO WHILE…LOOP, IF…END IFなどの構造化のためのステートメントをサポート。すっきりとした読みやすいプログラムが作成できます。

(2)プロシージャ

プログラムのビルディングブロック(組立部品)として、サブプログラムと関数という2つのプロシージャをサポート。一度作成したプロシージャを他のプログラムでも使えるために、開発効率が大幅にアップします。

(3)開発環境

Quick BASICは、インタプリタとコンパイラ、そして開発のためのツールからなる総合環境を提供します。インタプリタで確認したプログラムを、その場でコンパイルしてMS-DOSで単独に実行できます。

プログラミング作法

Quick BASICには、従来のBASICとの調和を保つために、行番号やGOTO文といった他のプログラミング言語にないか、あっても使われない機能が含まれています。

この2つ(行番号とGOTO文)を使うと、一見楽にプログラムを組むことができますが、他の言語には応用がきかないばかりか、応用力アップのための妨げとなります。Quick BASICの特徴を生かし、構造化のためのステートメントとプロシージャを使った応用性の高いプログラミング作法を身につけてください。

プログラミング言語の種類

コンピュータに対して一連の処理を指示するものがプログラムですが、このプログラムを書き表すための体系的な言葉(言語)が、プログラミング言語です。コンピュータの歴史とともに、プログラミング言語の歴史もあります。数多くの言語が開発されています。パーソナルコンピュータ向けの言語としては、次のものが有名です。

BASIC	汎用性の高い言語
C	汎用性の高い言語
COBOL	事務計算用の言語
FORTRAN	科学技術計算用の言語
LISP	人工知能用の言語
PROLOG	人工知能用の言語

汎用性の高い、何でも一応はできる言語として、BASICとCがあります。Cに追いやられた感のあるBASICですが、Quick BASICの出現により、その地位を挽回するでしょう。

使用機器

Quick BASIC は、日本電気の PC-9801 と各社の AX パソコンに対応しています。
本書は、PC-9801 と AX パソコンの両機器によって、その動作を確認しています。

使用機器1 PC-9801 VX21（日本電気）

使用機器2 AX パソコン（キヤノン AXi DX-20）

キーの表記

PC-9801とAXパソコンでは、キーのトップに書かれている記号に違いがあります。キーの名称がまったく異なるものについては、そのつどPC-9801とAXパソコンを分けて表記しますが、以下のキーについてはその類推が容易ですので、PC-9801のキートップに従って表記します。

PC-9801の表記にしたがうキー

PC-9801	AXパソコン
BS	Back Space
DEL	Delete
CTRL	Ctrl
SHIFT	Shift
TAB	Tab
↵	Enter
f·1	F 1
⋮	⋮
f·10	F 10
ESC	Esc
HOME CLR	Home

それぞれに表記するキー

PC-9801	AXパソコン
GRPH	Alt

システムの構成

システムの構成

Quick BASIC に含まれる主なファイルは次のとおりです。ざっと目を通しておいてください。

ファイル名	役　割
QB.EXE	Quick BASIC インタプリタ
QB.HLP	Quick BASIC ヘルプファイル
BC.EXE	ベーシックコンパイラ
LINK.EXE	リンカ
LIB.EXE	ライブラリアン
BRUN42A.EXE	ランタイムモジュール
BRUN42A.LIB	ランタイムライブラリ
BCOM42A.LIB	スタンドアロンライブラリ

インタプリタとコンパイラ

プログラミング言語は、人間の意志をコンピュータに伝達するための言葉です。いわゆる通訳の役目を果たすのが、インタプリタとコンパイラです。インタプリタは、同時通訳に相当します。プログラムの命令を1行ごとに翻訳して実行します。コンパイラは、全部聞き取ってからまとめて通訳するタイプです。コンパイラが通訳することを、コンパイルするといいます。

短いプログラムを作っては試しに実行するという試行錯誤的段階ではインタプリタが、完成されたプログラムを繰り返し実行する場合は、コンパイルした方が有利です。コンパイルした方が実行速度が速くなります。

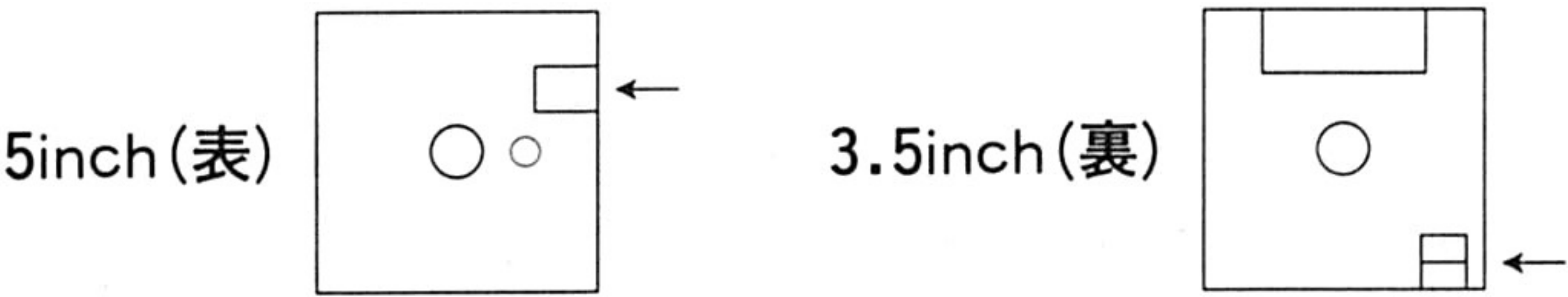
Quick BASIC は、プログラムの実行環境としてインタプリタとコンパイラの両方を備えています。EXE ファイルの作成と実行では、プログラムをコンパイルして実行することになります。

バックアップ

購入したQuick　BASICのディスクは、万一の破損に備えて必ずバックアップをとっておきましょう。ここでは、Quick　BASICのバックアップの手順を解説します。MS-DOSの基本的な知識を前提としています。

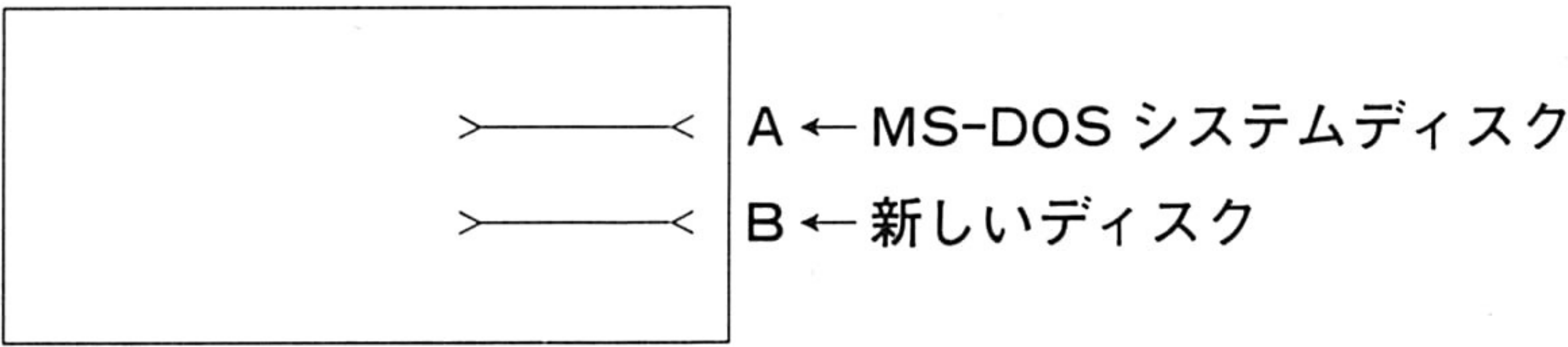
ライトプロテクト

まず、オリジナルのシステムディスクにライトプロテクト(書き込み禁止処理)をかけてください。3.5inchの場合は、ノッチを下げてください。5inchの場合は、ライトプロテクトシールを貼ってください。

5inch(表)　　3.5inch(裏)

ディスクのフォーマット

まず、新しいかその内容がなくなってもかまわないディスクを2枚とMS-DOSのシステムディスクを用意してください。この2枚のディスクをフォーマットします。フォーマットするためのMS-DOSのコマンドは、FORMAT.COMです。MS-DOSのシステムディスクをAドライブに入れて、リセットしてMS-DOSを立ち上げてください。

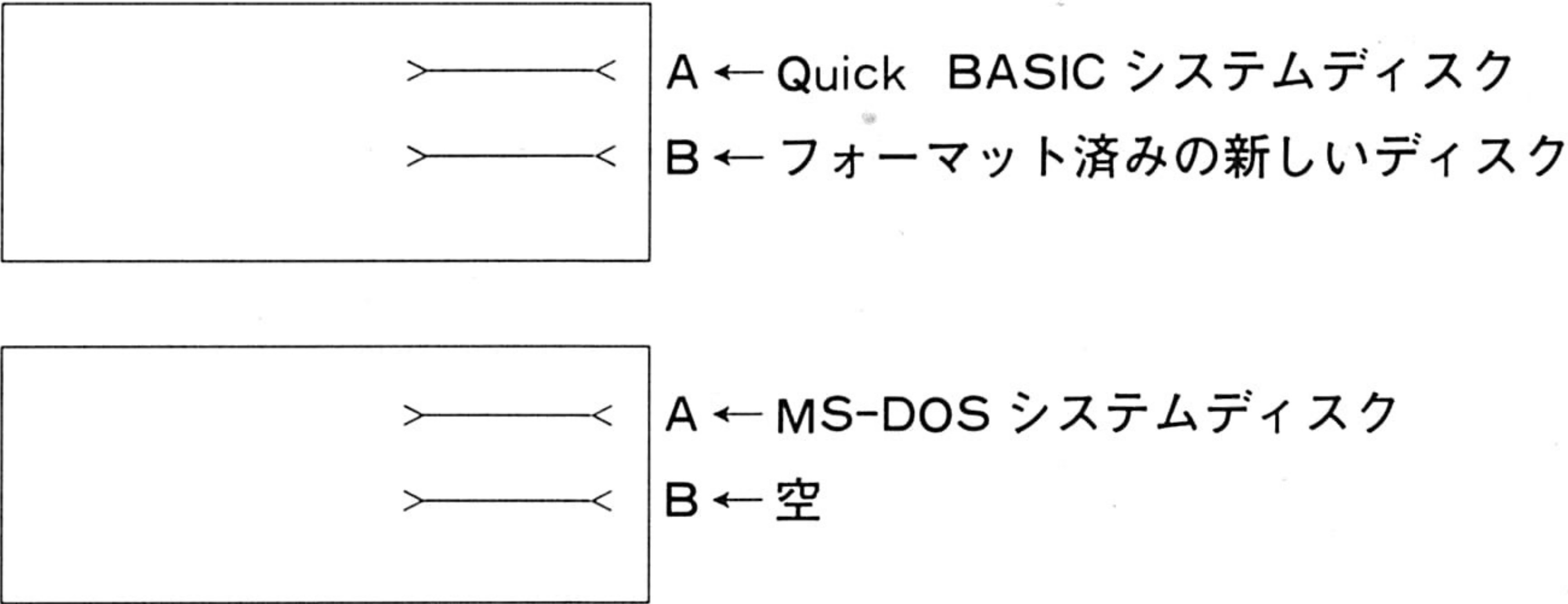
A>FORMAT　B:　/S⏎と入力して、フォーマットコマンドを立ち上げます。画面の指示に従い新しいディスクを挿入して⏎を押してください。

A←MS-DOSシステムディスク
B←新しいディスク

この作業を2回繰り返して、フォーマット済みのディスクを2枚作ります。

ディスクコピー

次に、Quick　BASICのシステムディスクを丸ごとコピーします。ディスクを丸ごとコピーするMS-DOSのコマンドは、DISKCOPY.COMです。

A>DISKCOPY　A:　B:⏎と入力してDISKCOPYコマンドを立ち上げたあと、画面の指示に従ってAドライブに送り側のQuick　BASICのディスクを、Bドライブに受け側の新しいディスクを挿入して⏎を押してください。

A←Quick　BASIC システムディスク
B←フォーマット済みの新しいディスク

A←MS-DOS システムディスク
B←空

Quick　BASIC のシステムディスクの枚数だけ、この操作を繰り返してください。

最後に、MS-DOS のシステムディスクをAドライブに戻して⏎を押してください。

保管

バックアップをとったマスタディスクは、安全な場所に保管して、このバックアップ品を使用して Quick　BASIC のインストールを行なってください。インストールの方法は、次のセクションで解説します。

DISKCOPY と COPY　*.*

MS-DOS は、すべてのファイルをコピーするコマンドが2つあります。ひとつは DISKCOPY で、もうひとつは COPY　*.* です。DISKCOPY は、ディスクをそのままコピーしますので、コピー元のディスクとコピー先のディスクはまったく同じものになります。一方 COPY　*.* コマンドは、ディスクの中をファイルごとにコピーしますので、コピー元のディスクとコピー先のディスクでは、その配置が異なります。

また、COPY　*.* コマンドはサブディレクトリを超えてコピーしませんのでコピー元とコピー先で正しくサブディレクトリを対応させながらコピーする必要があります。サブディレクトリを自動的に作成する機能はありません。DISKCOPY では、サブディレクトリのことは考えないですみます。

DISKCOPY コマンドは、コピー元が空のファイルでも実行されますので、コピー元とコピー先を間違えると、すべてを失うことになります。

インストール

インストール

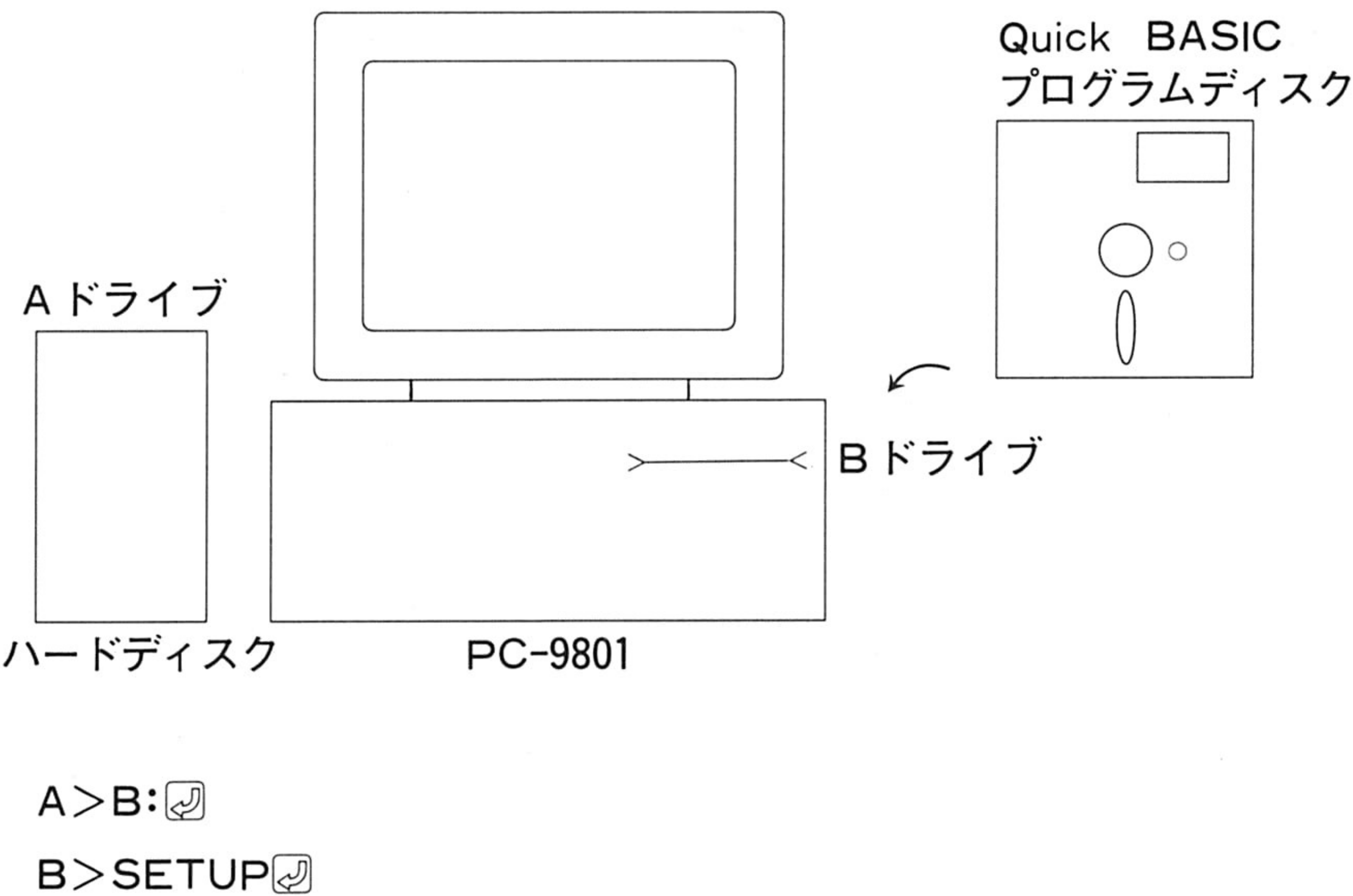
　自分の使うパソコンに、特定のシステムを組み込んで使えるようにすることを、そのシステムをインストールするといいます。ここでは、これからのパソコンの標準的な構成として、PC-9801＋ハードディスク に Quick　BASIC をインストールする手順を解説します。他の機種でも、ほぼ同様にしてインストールを行なうことができます。

　MS-DOS はハードディスクから立ち上がっているもの、すなわちハードディスクがAドライブとします。

セットアッププログラム

　Quick　BASIC には、インストールを行なうためのプログラム（セットアッププログラム）がついています。このセットアッププログラムを使ってハードディスクへのインストールを行なってみましょう。ディスクを次のようにセットして、セットアッププログラムを起動させます。

Quick　BASIC
プログラムディスク
Aドライブ
Bドライブ
ハードディスク
PC-9801

```
A＞B:↵
B＞SETUP↵
```

インストールの実行

```
セットアップを開始しますか？ [Y]
ハードディスクにセットしますか？ [Y]
Quick BASIC をどのドライブにインストールしますか？ [A]
Quick BASIC をどのディレクトリに転送しますか？ [Q][B]
数値演算エミュレートライブラリを転送しますか？ [N]
マウスを使用しますか？ [Y]
マウスの種類を選択してください ［1：バス／2：シリアル］ [1]
マウスドライバを選択してください ［1：MOUSE.SYS／2：MOUSE.COM］ [2]
上記の設定でよろしいですか？ [Y]
ドライブBにプログラムディスクをセットしてください。 [↵]
作業中です。しばらくお待ちください。

      ⋮

ドライブBにライブラリディスクをセットしてください。 [↵]
作業中です。しばらくお待ちください。
      ⋮
B>A:[↵]
A>
```

日本語の使用

フロントエンドプロセッサ

Quick BASIC で日本語を使用するためには、使用する日本語フロントエンドプロセッサの辞書の組み込みと CONFIG.SYS へのシステム登録が必要です。詳しくは、お使いになる日本語フロントエンドプロセッサのマニュアルを参照してください。

ここでは、ワードプロセッサ「一太郎」バージョン3に付いている、フロントエンドプロセッサ ATOK6 を組み込む例を解説します。

(1)辞書などのコピー

「一太郎」のシステムディスクから、辞書など必要なファイルをハードディスクにコピーしてください。

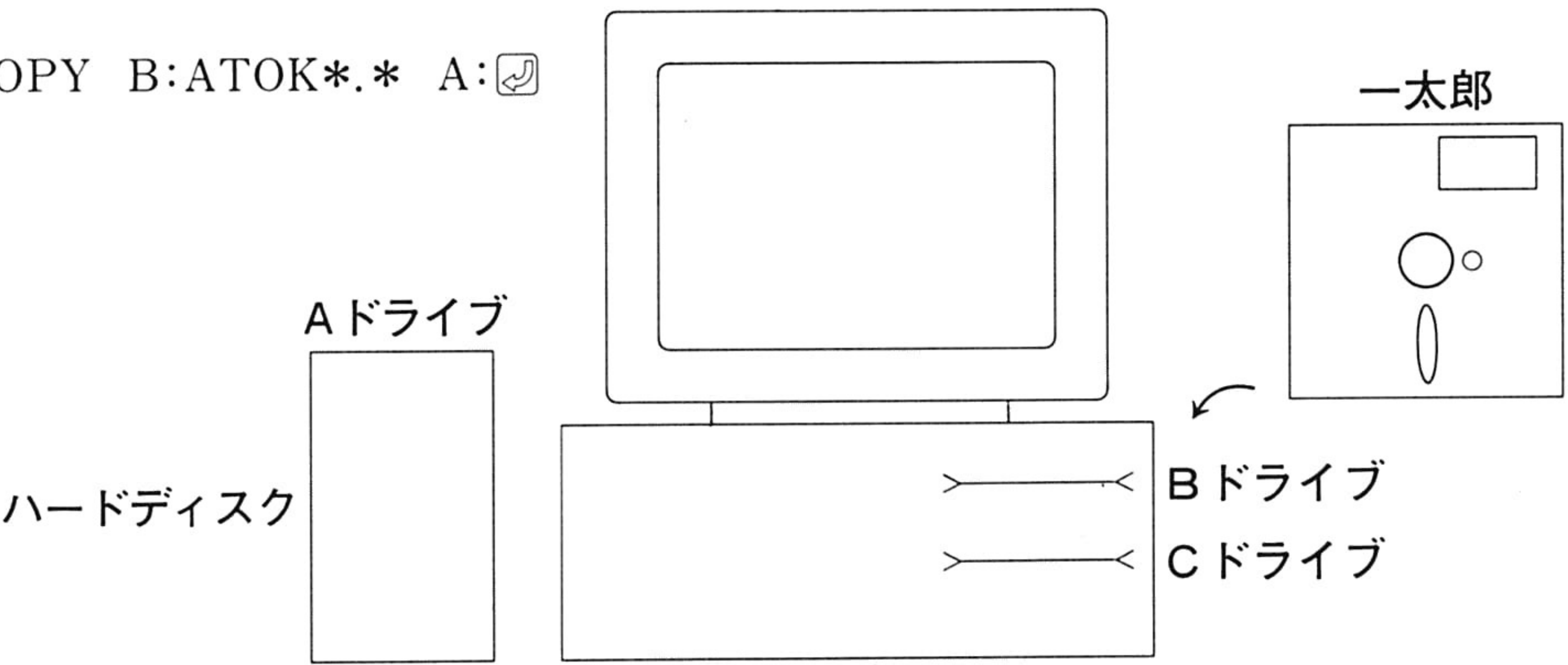
```
A>COPY  B:ATOK*.*  A:↵
```

一太郎

Aドライブ

ハードディスク

Bドライブ

Cドライブ

(2)CONFIG.SYS の作成

CONFIG.SYS を次のように作成してください。

CONFIG.SYS の作成

```
A>COPY  CON  CONFIG.SYS ↵
FILES  =  20 ↵
BUFFERS  =  20 ↵
DEVICE  =  ATOK6A.SYS ↵
DEVICE  =  ATOK6B.SYS ↵
CTRL + Z ↵
A>
```

NEW-CONF.SYS

CONFIG.SYS を作成する別の方法を解説します。まず、セットアップ時にディレクトリ QB にできる NEW-CONF.SYS というファイルをルートディレクトリにコピーしてください。エディタを用いて DEVICE ＝ ATOK6A.SYS と DEVICE ＝ ATOK6B.SYS の 2 行を挿入してください。そしてこの NEW-CONF.SYS を CONFIG.SYS としてセーブしてください。

CONFIG.SYS と AUTOEXEC.BAT

CONFIG.SYS は、MS-DOS が立ち上がるときに実行される MS-DOS 使用環境を設定するファイルです。このファイルの中には、使用する日本語環境として日本語フロントエンドプロセッサの名前を書きます。

AUTOEXEC.BAT は、MS-DOS が立ち上がった後一番最初に実行されるコマンドを書き連ねたファイルです。コマンドを探す順番を示す PATH や、マウスの使用を決めるための MOUSE の ON/OFF などを書いておきます。

パスの設定

NEW-PATH.BAT

Quick　BASIC を立ち上げるためには、Quick　BASIC のあるディレクトリへのパスを設定しておく必要があります。このパスを設定しておくと、ファイルの設定に際していちいちディレクトリを移動しなくてもすみます。立ち上げ時に自動的に Quick BASIC へのパスを設定するための AUTOEXEC.BAT の原型が、インストールの時にディレクトリ QB に NEW-PATH.BAT として作成されています。この NEW-PATH.BAT をルートディレクトリに AUTOEXEC.BAT というファイル名でコピーすることによって、AUTOEXEC.BAT を作成してください。

AUTOEXEC.BAT の作成

```
A>COPY  ￥QB￥NEW-PATH.BAT  AUTOEXEC.BAT↵
```

AUTOEXEC.BAT

作成された AUTOEXEC.BAT の内容は次のとおりです。MS-DOS の TYPE コマンドで確認してください。

AUTOEXEC.BAT の内容

```
A>TYPE  AUTOEXEC.BAT ↵
SET  COMSPEC=A:￥COMMAND.COM
SET  PATH=A:￥QB￥BIN
SET  LIB=A:￥QB￥LIB
MOUSE
```

MOUSE

AUTOEXEC.BAT の最後の1行は、マウスをオンにするコマンドです。

環境変数

PATH…………実行コマンドを探す順路

COMSPEC……MS-DOS の本拠地への順路

LIB ……………Quick　BASIC のライブラリへの順路

立ち上げ

インストールの終わったQuick BASICを立ち上げてみましょう。一度コンピュータをリセットしてください。Quick BASICを立ち上げるためには、MS-DOSのプロンプトに対して実行可能ファイル名QBを入力してください。

【例題1】 Quick BASICの立ち上げ QB

> Quick BASICを立ち上げてください。Quick BASICの実行可能ファイル名はQB.EXEです。

解答例

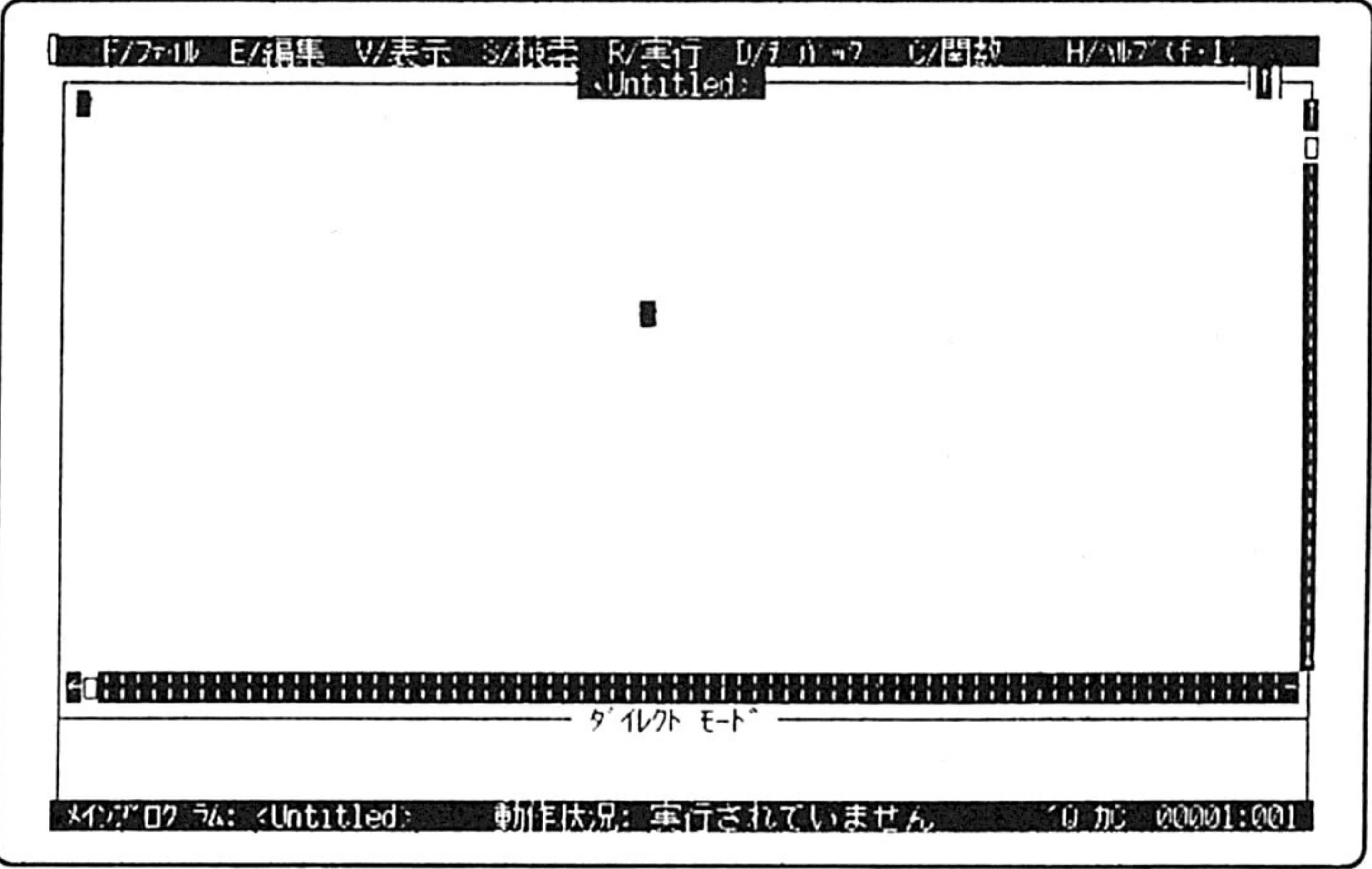
```
A>QB ↵
```

実行例

```
 F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ(f・1)
                    <Untitled>



                 ダイレクト モード

メインプログラム: <Untitled>   動作状況: 実行されていません   00001:001
```

マウスカーソル

編集画面の中にマウスカーソルがあります。マウスをインストール(使えるように)したものの「うるさいな」と感じたら、AUTOEXEC.BATの最後の1行(MOUSE)を削除して、リセットにより再び立ち上げてください。

終　了

終了メニュー

Quick　BASIC を終了するためには、プルダウンメニュー [F/ファイル] から [X/終了] を選択します。選択項目に終了を含む [F/ファイル] メニューは、次のキー操作でオープンされます。

PC-9801	AX パソコン
GRPH, F	Alt, F

矢印キーで反転表示領域を上下に動かして、最後の [X/終了] を選択してください。立ち上げてから何らかの文字が入力されていると、終了に先立ってファイルを保存するかどうかのダイアログが表れます。この場合は TAB によって、[N/いいえ] に移ってから ↵ を押すか、あるいはキーボードから直接 N を入力して Quick　BASIC を終了してください。

【例題2】　Quick　BASIC の終了

Quick　BASIC を終了させてください。

解答例

PC-9801 シリーズ

GRPH
F
↓ を11回
↵

AX パソコン

Alt
F
↓ を11回
↵

解説

■エンドレス

矢印キーによる選択項目の移動は、エンドレスに行なうことができます。

プルダウンメニュー表示後 [↑] 1回で一番下の [X/終了] を選択することもできます。

■頭文字

矢印キーによる選択の代わりに、終了を示すメニューの先頭文字Xを入力してもQuick BASICを終了できます。

■間違ったら [ESC]

間違って開いたメニューを閉じるには、[ESC] を押してください。

■マウスによるメニューの選択

マウスを使用している場合は、マウスカーソルを [F/ファイル] のいずれかの文字に合わせて左ボタンをクリックしてください。プルダウンメニューが表れますので、[X/終了] をクリックしてください。

モード

Quick BASICには3つのモードがあります。

1）エディットモード…プログラムを記述するためのエディタが呼び出されているモード

2）実行モード…………プログラムが実行されているモード

3）ダイレクトモード…入力された文字をプログラムとしてその場で実行するモード

Quick BASICが立ち上がっている状態は、エディットモードでプログラムの記述待ちの状態になっています。起動画面の下に、

――― ダイレクトモード ―――

というのがあります。そちらにカーソルを移すと、入力したものが即実行されます。第2章では、この3つのモードを切り換えて、Quick BASICによる学習をします。

第2章　基本操作

第2章では、インタプリタ上でのプログラムの作成から構成までと、MS-DOS から直接実行可能なファイルへの変換(コンパイル)までをひととおり実習します。プログラムの内容については二の次として、Quick BASIC の基本的な操作を学んでください。

第2章の概要

■プログラムの記述

プログラムは、Quick BASIC のエディタを使って記述します。

■プログラムの実行

エディタで記述したプログラムは、SHIFT＋f･5 ですぐに実行できます。

■プログラムの保存

プログラムは、保存しなければQuick BASIC の終了とともになくなってしまいます。

■プログラムのクリア

新しいプログラムを作成、あるいは読み込むためには、現在のプログラムをクリアしなくてはなりません。

■プログラムの読み込み

ディスクに保存されているプログラムは、再び読み込んで実行・編集ができます。

■EXE ファイルの作成

Quick BASIC に備わっているコンパイラによって、EXE ファイルを作成できます。

■EXE ファイルの実行

EXE ファイルは、MS-DOS のプロンプトから直接起動できます。

■エラーとその対処

Quick BASIC では、プログラムの文法上のエラーがあると、修正のためのヒントが表示されます。

■MS-DOS の実行

Quick BASIC を終了させずに MS-DOS を実行させることができます。

■ダイレクトモード

Quick BASIC には、1行ごとにプログラムを実行するダイレクトモードがあります。

■オンラインヘルプ

Quick BASIC の使い方や、プログラムの中で使うステートメントや関数の使い方が分からなくなったら、オンラインヘルプがあります。

プログラムの記述

エディットモード

Quick BASICを再び立ち上げてください。次に簡単なプログラムを記述して実行してみましょう。Quick BASICが立ち上がった直後の状態は、プログラムを記述できるエディットモードです。この状態からプログラムを書き始めます。

【例題3】 プログラムの記述

自分の名前を表示するプログラムを記述してください。自分の名前を表示するプログラム(命令)は、次のとおりです。

```
PRINT "山田 太郎"
```

解答例

```
PRINT "山田 太郎"⏎
```

解説

■漢字の入力

漢字の入力は、日本語フロントエンドプロセッサの仕様に従います。本書のようにATOKシステムを採用した場合は、CTRL+XFERで日本語モードになります。

■小文字と大文字

PRINTなどのQuick BASICのステートメントは、大文字で書いても小文字で書いてもかまいません。小文字で書いた場合は、Quick BASICが1行解読するたびに大文字に直して表示します。

■⏎の前の修正

1行の入力を終わる以前(⏎を押す前)であれば、BS,DELキーなどで入力した文字の修正ができます。

■⏎の後の修正

1行の入力の終わりを示す⏎が入力されると、Quick BASICはそれが正しいかどうか文法上のチェックをします。スペルミスがあると、エラーメッセージが表示されます。

このときの対応は、[エラーとその対処]のセクションを参照してください。

プログラムの実行

ラン

自分の名前を表示する【例題3】のプログラムを実行してみましょう。プログラムの実行は、SHIFT＋f·5 です。プログラムを実行させることをランさせるといいます。

【例題4】　プログラムの実行　SHIFT＋f·5

自分の名前を表示するプログラムを実行してください。

解答例　　　　**実行例**

解答例		実行例
PRINT "山田 太郎"	→ SHIFT＋f·5	山田太郎 何かキーを押してください

解説　**■エディットモードへの復帰**

プログラムの実行画面からエディットモードに戻るためには、何かキーを押してください。

■プルダウンメニュー　[R/実行]

プログラムの実行は、プルダウンメニュー[R/実行]の中から[S/スタート]を選択することによってもできます。

PRINT "山田 太郎"	→ PC-9801 GRPH, R, S AX パソコン Alt, R, S またはマウスクリック	山田 太郎 何かキーを押してください

プログラムの保存

作成したプログラムを保存しておきましょう。プログラムの保存は、プルダウンメニュー[F/ファイル]から[S/保存]を選択します。ファイル名を入力する小窓（ダイアログ）が表れますので、EX005と入力してください。Quick BASICの標準形式ファイルEX005.BASに、プログラムが保存されます。ファイル名の指定では、拡張子.BASが自動的に付加されますので拡張子の指定は不要です。

【例題5】 プログラムの保存

自分の名前を表示するプログラムを、ファイルEX005.BASに保存してください。

解答例

PRINT "山田 太郎"	→ PC-9801 GRPH, F, S AXパソコン Alt, F, S	ファイル名 EX005 ⏎

解説

■拡張子.BAS

ファイルには、拡張子.BASが自動的に付加されます。

■Quick BASIC形式（標準ファイル）

保存されるファイルは、Quick BASICに独特なファイル形式ですから、MS-DOSのTYPEコマンドでは確認できません。

■テキストファイル

プログラムをMS-DOSのテキストファイルとして保存する場合は、保存に際してのファイル名指定のダイアログでその形式を[T/テキストファイル]としてください。

プログラムのクリア

次の新しいプログラムの記述に入る、あるいは別に保存してあるプログラムを読み込んで編集・実行するためには、いま画面に表れているプログラムをクリアしてください。

プログラムのクリアは、プルダウンメニュー [F/ファイル] から [N/新規] を選択します。

【例題6】　プログラムのクリア

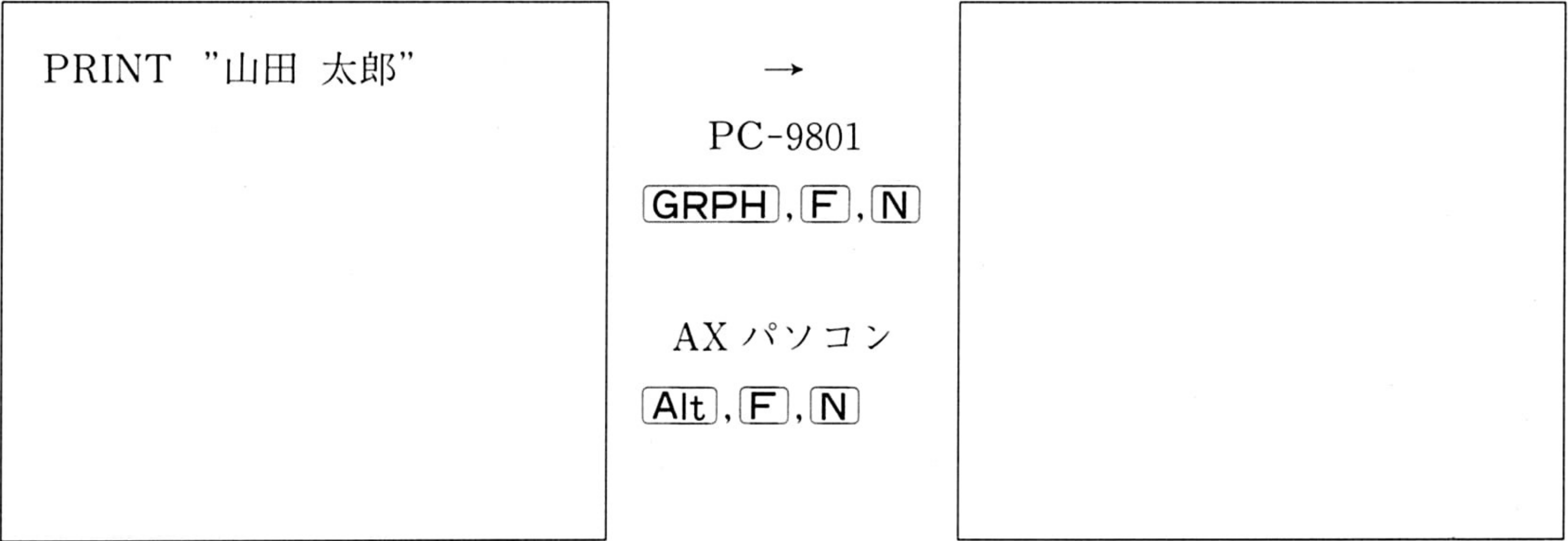
現在画面にあるプログラムをクリアしてください。

解答例

PRINT "山田 太郎"

→

PC-9801
GRPH, F, N

AXパソコン
Alt, F, N

解説　■ファイルへのセーブ

現在画面にあるプログラムがファイルへセーブされていないと、クリアする前にセーブするかどうかの問い合わせの画面が表れます。

プログラムの読み込み

自分の名前を表示するプログラムが保存されましたので、Quick BASIC をいったん終了後、プログラムを読み込んで実行してみましょう。

プログラムの読み込みは、プルダウンメニュー[F/ファイル]の中から[O/読み込み]を選択し、ファイル名の指定ダイアログに対して、ファイル名 EX005 を入力または選択します。ファイル名の入力では、拡張子.BAS は不要です。

【例題7】プログラムの読み込み

セーブしてある EX005.BAS ファイルを読み込んで実行してください。

解答例

→
PC-9801
GRPH, F, O
ファイル名 EX005⏎
AX パソコン
Alt, F, O
ファイル名 EX005⏎

```
PRINT "山田 太郎"
```

↓ SHIFT + f·5

```
山田太郎

何かキーを押してください
```

解説　**■ファイル名の選択**

ファイル名は、ファイル名ダイアログから選択することもできます。TAB キーによってファイル名ダイアログに移動し、↓↑と⏎で選択してください。

EXEファイルの作成

EXEファイル

Quick　BASICで作成したプログラムは、Quick　BASICに備わっている一定の手続き―コンパイルとリンク―を行なうことによって、MS-DOSのプロンプト状態から直接実行できるプログラムになります。このMS-DOSから直接実行可能なファイルの拡張子は.EXEになります。この拡張子にちなんで、MS-DOSから直接実行できるファイルをEXEファイルといいます。

プログラムのコンパイルは、プルダウンメニュー[R/実行]から、[X/EXEファイル作成]を選択します。ファイル名の指定と作成形式を入力するダイアログが表れますので、ファイル名EX005.EXEとファイル形式「ランタイム分離型EXEファイル」を選択して、EX005.EXEファイルを作成してください。

【例題8】　EXEファイルの作成

自分の名前を表示するプログラムEX005をコンパイルしてください。

解答例

PC-9801	AXパソコン
GRPH R X ファイル名　EX005 ⏎ ランタイム分離型EXEファイル	Alt R X ファイル名　EX005 ⏎ ランタイム分離型EXEファイル

解説　■ランタイム分離型EXEファイル

ここで作成したランタイム分離型EXEファイルの実行には、ランタイムモジュールBRUN42A.EXEを必要とします。完成後のプログラムを他のシステムで動かす場合には、そのシステムにランタイムモジュールBRUN42A.EXEをコピーするか、ファイル形式を[A/独立型EXE]ファイルとして作成してください。後者では、プログラムのサイズが大きくなりますが、ランタイムモジュールBRUN42A.EXEのコピーは不要になります。独立型EXEファイルを作成する場合は、BCOM42A.LIBとのリンクが必要です。

EXEファイルの実行

それでは、作成したEXEファイルを実行してみましょう。MS-DOSのプロンプトに対して、EXEファイル名を入力することによって実行できます。

【例題9】 EXEファイルの実行

自分の名前を表示するプログラムEX005.EXEをMS-DOSから実行してください。

解答例

Quick BASICを終了する
A>EX005 ⏎

解説 **■作成後終了**

GRPH,R でEXEファイルを作成するときに、一番最後の指定項目として、[E/EXEファイル作成後終了]を選択しておくと、上の解答例にあるQuick BASICを終了するための手順が不要になります。EXEファイル作成終了後、Quick BASICを終了してMS-DOSになります。

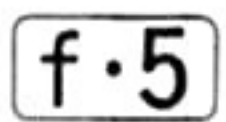
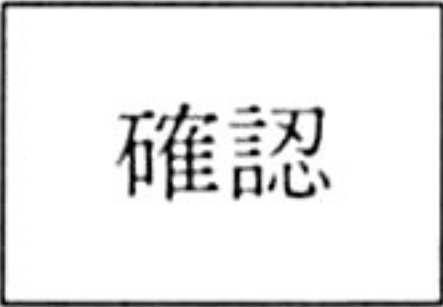
Quick BASICのEXEファイル

Quick BASICでは、次の2つのEXEファイルを作成できます。

(1) ランタイム分離型…実行に際して分離されているランタイムモジュールBRUN42A.EXEを必要とします。
リンクするライブラリは、BRUN42A.LIBです。

(2) 独立型EXEファイル…実行に際しては、何も必要ではありません。
リンクするライブラリは、BCOM42A.LIBです。

エラーとその対処

シンタックスエラー

Quick BASIC では、プログラムの構文上のエラー(シンタックスエラー)を、自動的に判別して適切なメッセージを出します。ここでは、わざと入力ミスして構文エラーを発生させ、その対処法を確認しておきましょう。

【例題10】 エラーとその対処

自分の名前を表示するプログラムを次のように作成した場合、エラーになります。対処して正しいプログラムに直してください。

PRUNT "山田 太郎" ⏎　　PRINT のスペルを間違った例

解答例

1) プログラムの実行　[SHIFT]+[f･5]

構文に誤りがあります(ERR=2)　←エラーメッセージ

[確認]

2) [ESC] または ⏎

3) 構文の誤り修正　　PRUNT → PRINT

エラーの種類

プログラムの作成時に発生するエラーには、次の3つのものがあります。

(1) 構文エラー(シンタックスエラー)

つづりの間違いなど文法上の誤りで、Quick BASIC がチェックしてくれます。

(2) 意味エラー(セマンティックエラー)

プログラマの思考が間違っているために、期待どおりに働かないエラー。

(3) 実行時エラー(ランタイムエラー)

プログラムを実行させてはじめて生じる、コンピュータのハードと密着したエラー。

MS-DOSの実行

Quick　BASICの実行中に、MS-DOSに移ってMS-DOSのコマンドを実行することができます。例えばディレクトリを確認したり、不要なファイルを削除することなどができます。

Quick　BASICを終了させることなくMS-DOSを実行するには、[F/ファイル] メニューの中から [D/MS-DOSコマンド] を選択します。

EXIT

MS-DOSコマンド実行の後Quick　BASICに戻ってくるには、MS-DOSのプロンプトに対してEXITと入力してください。

【例題11】　MS-DOSの実行

Quick　BASICを終了させることなくAドライブのディレクトリを確認してください。確認後、Quick　BASICに戻ってきてください。

解答例

PC-9801	AXパソコン
GRPH F D	Alt F D

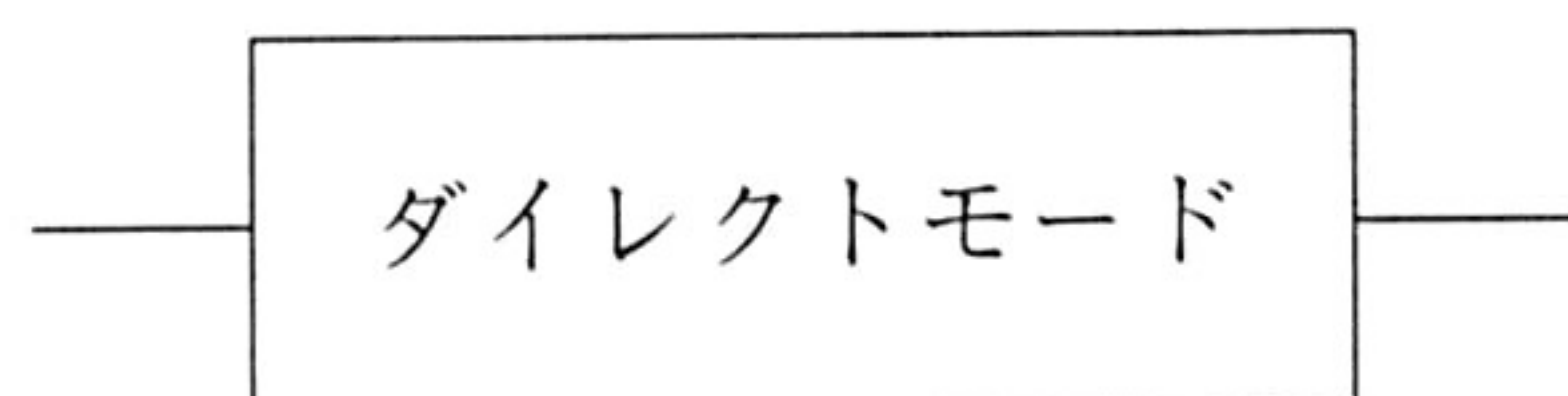
A>DIR ⏎
…
…
A>EXIT ⏎
何かキーを押してください

解説　■COMSPEC

Quick　BASICの立ち上げ時に、環境変数COMSPECが設定されていることが必要です。

ダイレクトモード

ダイレクトモード

エディットモードでの画面の下に、

―― ダイレクトモード ――

というのが見えますね。ここにQuick BASICの命令を書いて⏎をすると、SHIFT＋f·5を押さなくてもすぐに実行されます。上の広い方のエディット画面でプログラムを作成中に、

こんなことできるかな？

という、チョイ試しに使えます。

f·6

ダイレクトモードとエディットモードの間は、f·6を押すことによって行ったり来たりできます。

【例題12】　ダイレクトモード

ダイレクトモードで自分の名前を書くプログラムを実行してください。

解答例

f·6……カーソルがダイレクトモードの領域に移ります。
PRINT　"山田　太郎"⏎……すぐに実行されます。

解説　■もう一度　f·6

エディットモードに戻るために、もう一度f·6を押してください。

■1行だけ

ダイレクトモードで実行できるのは、1行だけです。1行で複数の命令をダイレクトモードで実行させることもできます。

PRINT　"山田　太郎":PRINT　"…"

のように、コロンで命令をつなげて1行にしてください。

これをマルチステートメントといいます。

オンラインヘルプ

Quick BASIC には、オンラインヘルプといって、マニュアルを開かなくてもいろいろなヘルプが得られるファイルが入っています。

全ヘルプ　[f･1]

ヘルプ全体をみる場合は、[f･1] を押してください。1画面ごとに読み取る形のヘルプが出ます。

個別ヘルプ　[SHIFT]+[f･1]　(キーワード)

特定のステートメントや関数についての詳細なヘルプが欲しい場合は、そのキーワードのいずれかにカーソルをおいて [SHIFT]+[f･1] を押してください。詳細なヘルプが出ます。

【例題13】　オンラインヘルプ

ここまでの例題に使用した PRINT とは、いったい何でしょうか。ヘルプを読んでください。

解答例

1．PRINT のいずれかにカーソルをおく
2．[SHIFT]+[f･1]
3．読み終わったら [ESC] を押す

第３章　操作マニュアル

第３章は、Quick　BASICの基本的な操作をマニュアルとしてまとめてあります。操作方法が不明の場合に参照してください。

なお、コマンドのいろいろな選択方法について、２、３の例題がありますので、確認のため実習しておいてください。

第３章の概要

■コマンドの選択

コマンドの選択には、３つの方法があります。

■コマンド選択一覧表

各コマンドの選択方法を、見やすい表形式にまとめておきました。

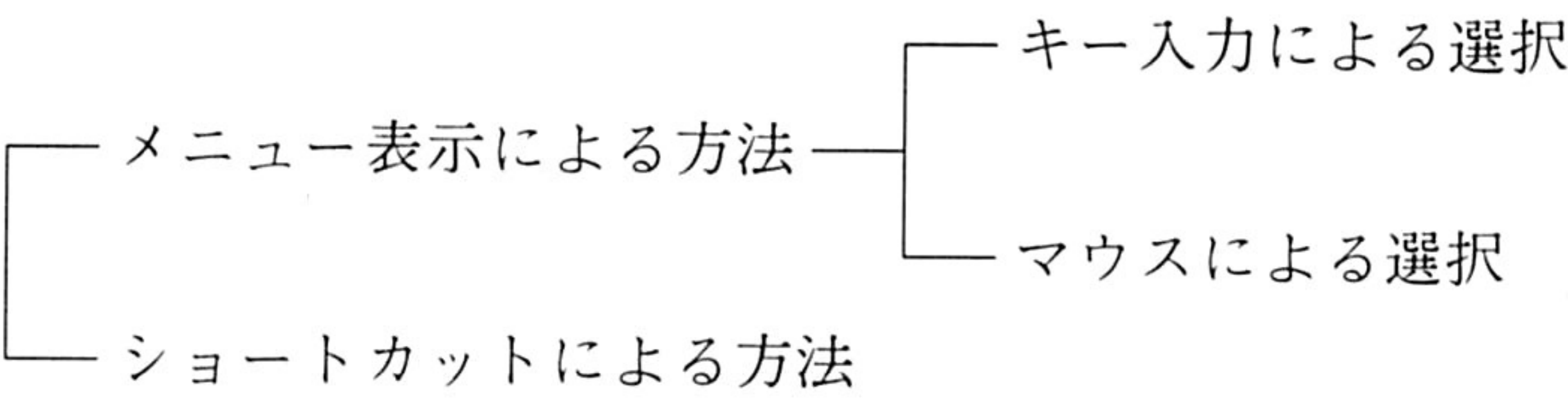
■エディットモード

エディットモードでのカーソルの動きなどをまとめておきました。

コマンドの選択

Quick BASIC では、ファイルからの読み込みやプログラムの実行などの機能の選択を、コマンドの選択といいます。コマンドの選択には、3つの方法があります。

コマンドの選択方法

- メニュー表示による方法
  - キー入力による選択
  - マウスによる選択
- ショートカットによる方法

(1)キー入力による選択

キー入力によるコマンドの選択ではまず [GRPH] キー(AX パソコンでは [Alt] キー)によりメニューバーをアクティブにしてから、対応するコマンド群の頭文字を入力、または [⇥][⇤] と [↵] キーで選択してプルダウンメニューを表示させます。表示されたプルダウンメニュー内のコマンドは、コマンドの頭文字を入力するか、[↓][↑] と [↵] キーで選択します。

【例題14】　コマンドの選択　＜キー入力による方法＞

> 自分の名前を表示するプログラムを、キー入力によってメニューを選択することによって実行してください。

解答例

PC-9801

[GRPH]
[R]
[S]

AX パソコン

[Alt]
[R]
[S]

(2)マウスによる選択

マウスによるコマンドの選択では、メニューバーのメニュー項目(コマンド群)を左ボタンでクリックしてください。プルダウンメニューが表れますので、該当するコマンドを左クリックすることによって選択してください。

【例題15】　コマンドの選択　<マウスによる方法>

自分の名前を表示するプログラムをマウスによってコマンドを選択し、実行してください。

(3)ショートカットによる方法

Quick BASIC では、ひんぱんに使うコマンドをファンクションキーの組み合せなどにより、より速く選択できるようになっています。例えば、プログラムの実行は、メニュー選択では [GRPH],[R],[S] (AX パソコン：[Alt],[R],[S]) ですが、[SHIFT]＋[f･5] で同じことができます。ショートカットは、プルダウンメニューの右に書かれています。

【例題16】　コマンドの選択　<ショートカットによる方法>

自分の名前を表示するプログラムをショートカットによってコマンドを選択し、実行してください。

解答例

[SHIFT]＋[f･5]

コマンド選択一覧表

Quick BASIC のコマンド体系の全体を次に示します。

メニュー	主な機能
F/ファイル	ファイルの作成、読み込みなど
	Quick BASIC の終了
E/編集	編集時のカット&ペーストなど
V/表示	ファイルの中の表示単位の選択
	画面の色などの設定
S/検索	検索や置換など
R/実行	プログラムの実行
	EXE ファイルの作成
D/デバッグ	ブレークポイントの設定やトレース機能など
C/関数	プロシージャの親子関係の表示
H/ヘルプ	Quick BASIC のオンラインヘルプ

F/ファイルメニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
N/新規	新しいプログラムを作成するために、エディタをクリアします。		
O/読込	すでにあるディスク上のプログラムを読み込みます。		
M/追加	現在のモジュールにファイルを追加します。		
S/保存	現在のモジュールをディスク上のファイルに保存します。		
A/名前を変えて保存	現在のモジュールを別名・別形式で保存します。		
V/すべて保存	メモリ上のすべてのモジュールをディスクファイルに保存します。		
C/サブファイル作成	新しいモジュール、インクルードファイルまたはドキュメントファイルを作成します。		
L/サブファイル常駐	モジュール、インクルードファイルまたはドキュメントファイルを読み込んで常駐させます。		
U/サブファイル解放	読み込まれているモジュールやインクルードファイルまたはドキュメントファイルをメモリから解放します。		
P/印刷	指定されたテキストまたはモジュールを印刷します。 **印刷の対象** S/指定範囲 W/アクティブウィンドウ M/カレントモジュール A/全体		
D/MS-DOS コマンド	一時的に Quick BASIC から抜けて MS-DOS に移ります。		
X/終了	Quick BASIC を終了します。		

E/編集メニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
U/元にもどす	最後の編集動作を取り消して、その前の状態に戻します。	GRPH＋BS	Alt＋Back Space
T/カット	テキストをクリップボードに格納し、そのテキストを削除します。	SHIFT＋DEL	Shift＋Delete
C/コピー	テキストをクリップボードに格納します。テキストはそのままです。	CTRL＋INS	Ctrl＋Insert
P/ペースト	クリップボードの内容を、現在のカーソル位置に貼り付けます。	SHIFT＋INS	Shift＋Insert
E/削除	テキストを削除します。クリップボードへは格納しません。	DEL	Delete
S/新規SUB…	新しいSUBプロシージャ定義用のウィンドウを開きます。 ●プロシージャ名を指定します。 N/名前：		
F/新規FUNCTION	新しいFUNCTIONプロシージャ定義用のウィンドウを開きます。 ●プロシージャ名を指定します。 N/名前：		
Y/構文チェック	エディタの自動構文チェック機能をオンまたはオフにします。		

V/表示メニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
S/SUB一覧…	読み込まれている SUB, FUNCTION, モジュール, インクルードファイル, またはドキュメントファイルを一覧形式で表示します。 ↓ ↑ で選択し ⏎ を押すことによってそのモジュールを編集対象として選択できます。	f·2	F2
E/次のSUB	次のサブプログラムを表示します。	SHIFT＋f·2	Shift＋F2
P/画面分割	画面の分割をオンまたはオフにします。		
N/次のステートメント	次のステップで実行されるステートメントを表示します。		
U/実行画面	編集画面から出力画面に切り換えます。	f·4	F4
I/インクルードファイル編集	インクルードファイルを編集します。		
L/インクルードファイル表示	インクルードファイルを表示します。		
O/オプション	表示属性を変更します。 **テキストの表示属性** 文字　B/黒色, B/青色, G/緑色, C/水色, 　　　R/赤色, M/紫色, Y/黄色, W/白色 背景　B/黒色　＊色の選択は ← → キーによる 　　　H/反転 **カレントステートメントの表示属性** 文字　B/黒色, B/青色, G/緑色, C/水色, 　　　R/赤色, M/紫色, Y/黄色, W/白色 背景　B/黒色 　　　H/反転 **ブレークポイント行の表示属性** 文字　B/黒色, B/青色, G/緑色, C/水色, 　　　R/赤色, M/紫色, Y/黄色, W/白色 背景　B/黒色 　　　H/反転 **全体の表示オプション** S/スクロールバー T/タブ間隔：		

S/検索メニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
F/検索…	指定したテキストを検索します。 **テキストの指定** F/検索文字列 **検索範囲** 1/アクティブウィンドウ 2/カレントウィンドウ 3/全体 **限定条件** M/大小文字区別 W/全体一致		
S/指定文字列検索	プログラムの中から選択したテキストを検索します。 **テキストの指定** F/検索文字列 **検索範囲** 1/アクティブウィンドウ 2/カレントウィンドウ 3/全体 **限定条件** M/大小文字区別 W/全体一致	CTRL+¥	Ctrl+¥
R/次を検索	指定したテキストでの検索を続行します。	f·3	F3
C/置換	指定したテキストを検索して、新しいテキストに置換します。 **テキストの指定** F/置換前 **新しく置き換えるテキストの指定** T/置換後 **検索する範囲** 1/アクティブウィンドウ 2/カレントウィンドウ 3/全体	CTRL+Q+A	Ctrl+Q+A

C/置換	**限定条件** M/大小文字区別 W/全体一致 **置換方式** V/逐次確認 C/一括置換	CTRL＋Q＋A	Ctrl＋Q＋A
L/ラベル	ラベルの検索を行ないます。 **ラベルの指定** F/検索文字列 **検索範囲** １/アクティブウィンドウ ２/カレントウィンドウ ３/全体 **限定条件** M/大小文字区別 W/全体一致		

R/実行メニュー

		ショートカット	
コマンド	機　能	PC-9801	AX パソコン
S/スタート	現在のプログラムを先頭から実行します。	SHIFT+f·5	Shift+F5
R/リスタート	現在のプログラムを実行します。		
N/続行	現在のプログラムの中断を解いて実行を続けます。	f·5	F5
C/COMMAND$入力 …	COMMAND$ 関数が返す文字列を設定します。		
X/EXE ファイル作成…	実行可能ファイルを作成します。 **作成するファイル名** N/EXE ファイル名： **EXE ファイルの形式** X/ランタイム分離型 EXE ファイル A/独立型 EXE ファイル **デバッグコードの設置** D/デバッグコード **ファイル作成後の状態** M/EXE ファイル作成 E/EXE ファイル作成後終了		
L/ライブラリ作成…	クイックライブラリを作成します。 **作成するライブラリ名** N/Quick ライブラリファイル名： **デバッグコードの設置** D/デバッグコード **ライブラリ作成後の状態** M/ライブラリ作成 E/ライブラリ作成後終了		
M/メインモジュールの設定…	メインモジュールを変更します。		

D/デバッグメニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
A/ウォッチの追加…	ウォッチ式の追加設定。		
W/ウォッチポイント…	ウォッチポイントを設定します。		
D/ウォッチの削除…	ウォッチ式を個別に解除します。		
L/全ウォッチ削除	ウォッチ式をすべて解除します。		
T/トレース	実行中のステートメントを強調します。		
H/ヒストリ	ステートメントが実行された順序を記録します。		
B/ブレークポイント	カーソル位置を、ブレークポイントとして設定または解除します。	f·9	F9
C/全ブレークポイントの解放	すべてのブレークポイントを解除します。		
S/カレントステートメントの設定	次に実行するステートメントを指定します。		

C/関数メニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
M/<PROCEDURE>	現在呼び出されているプロシージャ(<PROCEDURE>)を呼び出しているプロシージャ名の位置にカーソルを移動します。		

H/ヘルプメニュー

コマンド	機　　能	ショートカット	
		PC-9801	AX パソコン
G/全般…	ヘルプ情報全般を表示します。	f･1	F1
T/キーワード	カーソル位置のBASICのキーワードについてのヘルプ情報を表示します。	SHIFT+f･1	Shift+F1
C/ヘルプを閉じる	ヘルプを終了します。	ESC	Esc

エディットモード

Quick BASICのエディットモード(プログラムを作成するモード)におけるカーソルの移動など、編集のためのコマンドを以下にまとめておきます。

カーソルの移動	PC-9801	AXパソコン	コントロール系
1文字分左に移動	←	←	CTRL+S
1文字分右に移動	→	→	CTRL+D
1単語分左に移動	CTRL+←	Ctrl+←	CTRL+A
1単語分右に移動	CTRL+→	Ctrl+→	CTRL+F
1行上に移動	↑	↑	CTRL+E
1行下に移動	↓	↓	CTRL+X
第1インデントレベルまで移動	HOME CLR	Home	
その行の先頭に移動			CTRL+Q+S
次の行の先頭に移動	CTRL+↵	Ctrl+↵	CTRL+J
行の末尾に移動	HELP	End	CTRL+Q+D
ウィンドウの最上端に移動			CTRL+Q+E
ウィンドウの最下端に移動			CTRL+Q+X
モジュール／プロシージャの先頭に移動			CTRL+Q+R
モジュール／プロシージャの終わりに移動	CTRL+HELP	Ctrl+End	CTRL+Q+C
マーカ*の設定			CTRL+K+n
マーカに移動			CTRL+Q+n

(注)

マーカは、0～3までの4つを指定できます。nに0～3までの数字を入力してください。

挿入	PC-9801	AX パソコン	コントロール系
挿入モードのオン／オフの切り替え	INS	Insert	CTRL＋V
下に1行挿入	HELP＋↵	End＋↵	
上に1行挿入			HOME＋CTRL＋N
クリップボードの内容を挿入	SHIFT＋INS	Shift＋Insert	
タブをカーソル位置または指定した行の先頭に挿入	TAB	Tab	CTRL＋I

削除	PC-9801	AX パソコン	コントロール系
現在カーソルのある行をクリップボードに保存して削除			CTRL＋Y
カーソル位置から行の末尾までをクリップボードに保存して削除			CTRL＋Q＋Y
カーソルの左の1文字を削除	BS	Back Space	CTRL＋H
カーソル位置の文字を削除	DEL	Delete	CTRL＋G
単語を削除			CTRL＋T
指定したテキストをクリップボードに保存しないで削除	DEL	Delete	CTRL＋G
指定したテキストをクリップボードに保存して削除	SHIFT＋DEL	Shift＋Delete	

削除(つづき)	PC-9801	AX パソコン	コントロール系
指定した行のインデントを1レベル分（のスペース）削除	SHIFT + TAB	Shift + Tab	

コピー	PC-9801	AX パソコン	コントロール系
指定したテキストをクリップボードに保存してコピー	CTRL + INS	Ctrl + Insert	

スクロール	PC-9801	AX パソコン	コントロール系
1行上にスクロール	↑	↑	CTRL + W
1行下にスクロール	↓	↓	CTRL + Z
1画面分上にスクロール	ROLL DOWN	Page Up	CTRL + R
1画面分下にスクロール	ROLL UP	Page Down	CTRL + C
1画面分左にスクロール	CTRL + ROLL DOWN	Ctrl + Page Up	
1画面分右にスクロール	CTRL + ROLL UP	Ctrl + Page Down	

指定	PC-9801	AX パソコン	コントロール系
カーソル位置の左の1文字を指定	SHIFT + ←	Shift + ←	
カーソル位置の右の1文字を指定	SHIFT + →	Shift + →	
カーソルのある行を指定	SHIFT + ↓	Shift + ↓	
カーソルのある行の上の1行を指定	SHIFT + ↑	Shift + ↑	
カーソル位置から左の単語の始めまで指定	SHIFT + CTRL + ←	Shift + Ctrl + ←	

指定(つづき)	PC-9801	AX パソコン	コントロール系
カーソル位置から右の単語の終わりまで指定	SHIFT + CTRL + →	Shift + Ctrl + →	
カーソル位置から下に1画面分指定	SHIFT + ROLL DOWN	Shift + Page Up	
カーソル位置から上に1画面分指定	SHIFT + ROLL UP	Shift + Page Down	
カーソル位置から右に1画面分指定	SHIFT + CTRL + ROLL DOWN	Shift + Ctrl + Page Up	
カーソル位置から左に1画面分指定	SHIFT + CTRL + ROLL UP	Shift + Ctrl + Page Down	
モジュール／プロシージャの先頭まで		Shift + Ctrl + Home	
モジュール／プロシージャの終わりまで	SHIFT + CTRL + HELP	Shift + Ctrl + End	

第 2 部

プログラミングⅠ（初級編）

第1章　プログラミングの基礎

第１章では、プログラムの基本形とプログラムの中で使われる変数について学習しましょう。

第１章の概要

■プログラムの基本形

プログラムは、入力されたデータを処理して出力するものです。

■プログラムの実行順

プログラムは特に指示がなければ、その記述された順に上から下へ実行されます。

■変数

変数は、プログラムの中で処理の対象となるデータを一時的に蓄えておく器（うつわ）です。

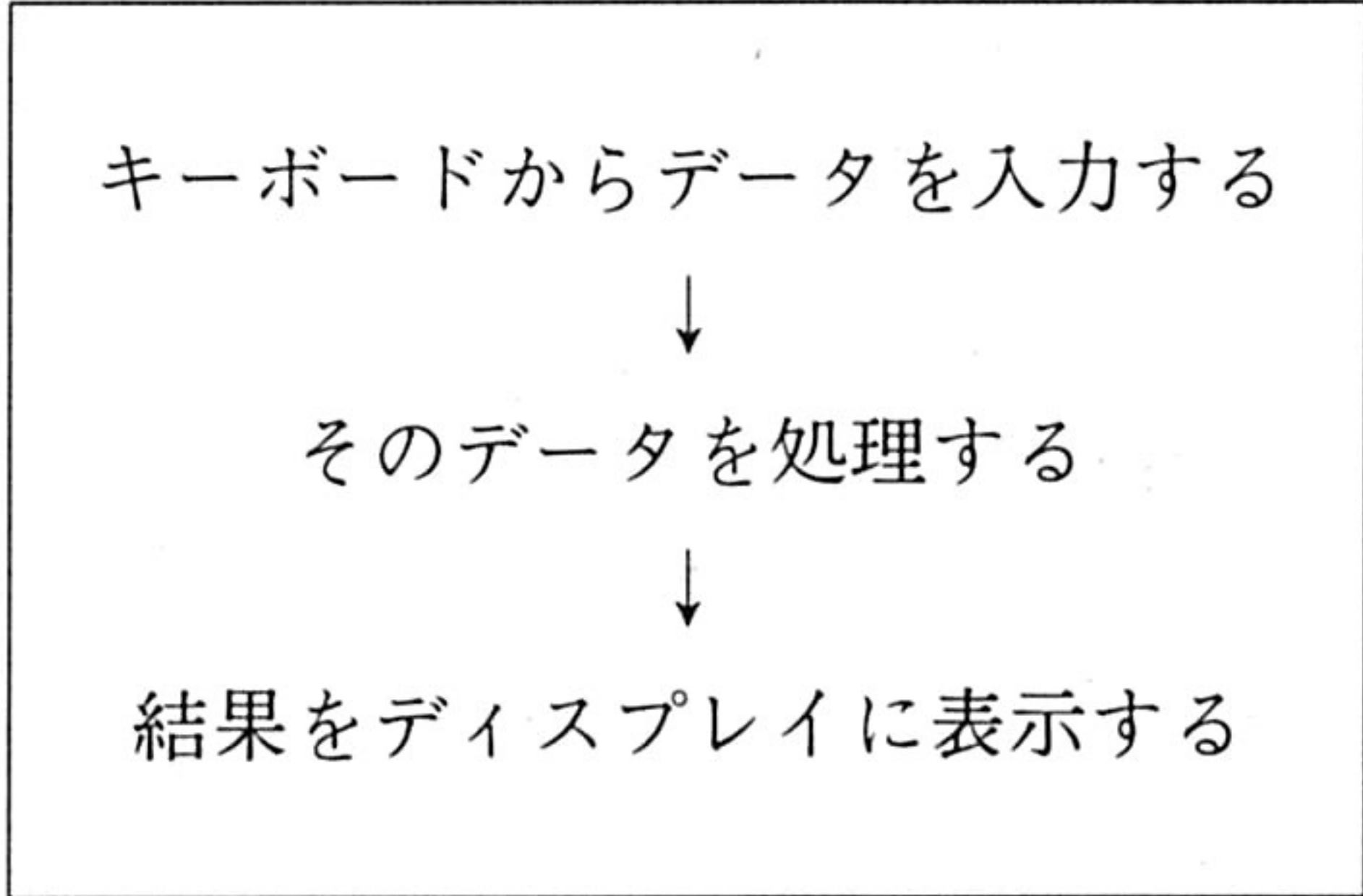
■データの型と変数の型

変数は、その中に蓄えられるデータの型に合ったものを用意する必要があります。

プログラムの基本形

プログラムの基本形、すなわちコンピュータの基本的な動作は次のとおりです。

プログラムの基本形

キーボードからデータを入力する
↓
そのデータを処理する
↓
結果をディスプレイに表示する

(プログラム例)　2つの数の和

キーボードから2つの数字を入力し、その合計を表示するプログラム

```
INPUT A          データ入力
INPUT B
C = A + B        データの処理
PRINT C          結果の表示
```

(注) INPUTやPRINTなどについてはあとで解説します。雰囲気として理解してください。

【例題17】　プログラムの基本形

上のプログラム(例)を作成し、実行してください。

解答例

```
INPUT A
INPUT B
C = A + B
PRINT C
```

```
?12 ⏎
?15 ⏎
 27
```

解説　■?マーク

INPUT文による入力待ちでは、?マークが表示されます。

プログラムの実行順

プログラムは、プログラムの中で特に指示をしなければ、上から下に順序よく行なわれます。

(プログラム例)　3つの数の平均の計算

```
INPUT  A
INPUT  B
INPUT  C
SUM  =  A  +  B  +  C
AVE  =  SUM  /  3          (割り算はスラッシュ/で表現します)
PRINT  AVE
```

【例題18】　プログラムの実行順

上のプログラム(例)を作成し、実行してください。

解答例

```
INPUT  A
INPUT  B
INPUT  C
SUM  =  A  +  B  +  C
AVE  =  SUM  /  3
PRINT  AVE
```

実行例

```
? 4 ⏎
? 6 ⏎
? 2 ⏎
  4
```

解説　**■四則演算**

四則演算には、それぞれ次の演算子を使います。

たし算　＋　，ひき算　－　，かけ算　＊　，割り算　／

■ステートメント

プログラムの流れを制御する指示(ステートメント)には、IF や WHILE，FOR などがあります。これらについては、順々に解説していきます。

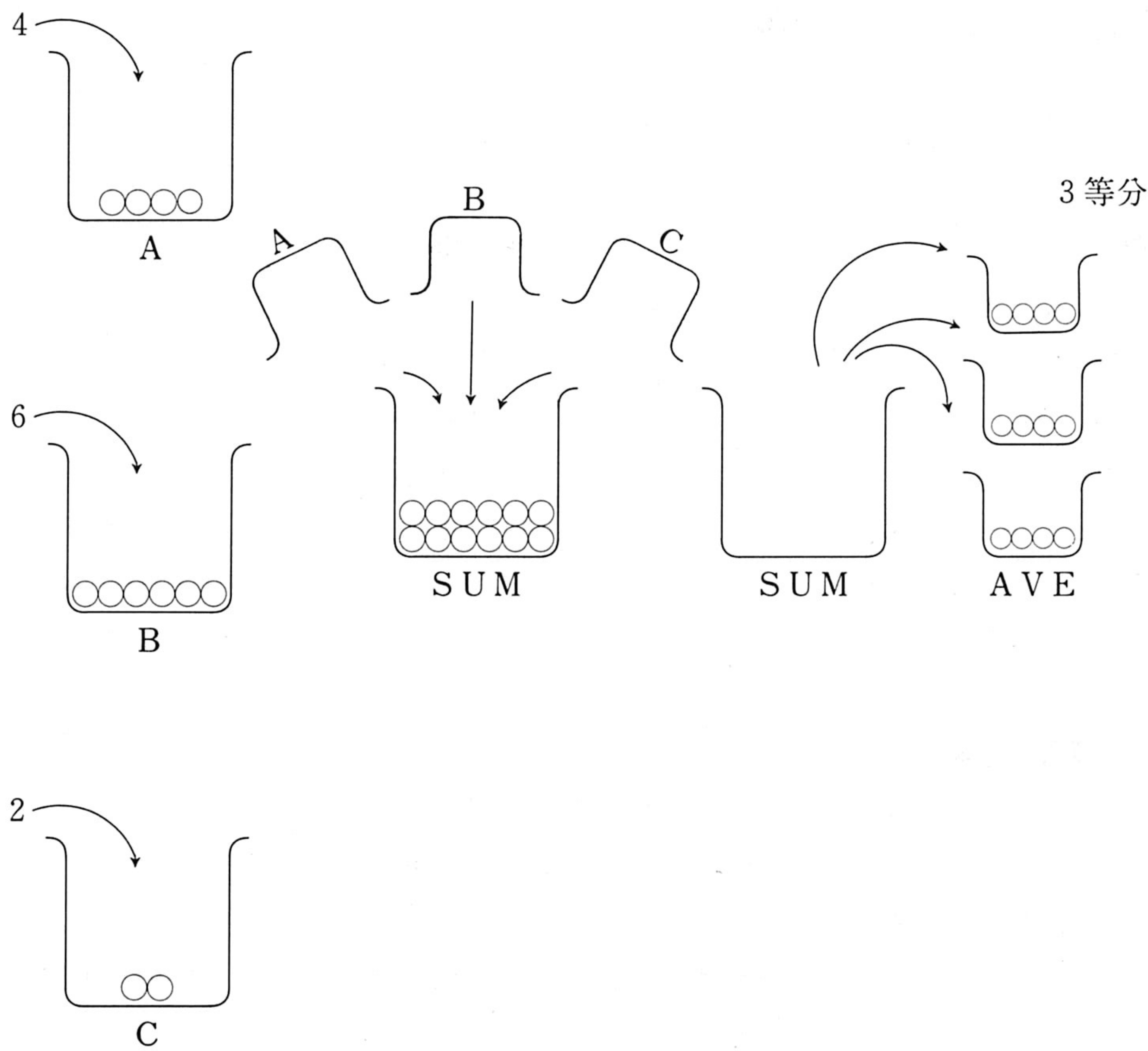
変数

変数

キーボードから入力されたデータをコンピュータの中で処理するためには、そのデータを入れる器(うつわ)が必要です。この器のことを、変数といいます。ここまでの例では、A，B，C，SUM，AVE などが変数です。

変数名

入ってくるデータひとつにひとつの器(変数)が必要ですから、プログラムの中では多くの器(変数)を使うことになります。器に名前をつけて区別することになります。この器(変数)につけられた名前を、変数名といいます。

先ほどの3つの数の平均を計算する問題を考えてみましょう。変数(器)を中心に考えると次のようになります。

変数名の制限

変数名を付けるときは、その性質をイメージできる4文字程度の英字のつづりがよいでしょう。なお、変数名の付け方には次の制限があります。

変数名の制限

(1)長さ………40文字まで

(2)文字種……変数名として使える文字は、

(a)アルファベット

(b)アラビア数字

(c)小数点

(d)%，&，!，#，$

ただし、(c)小数点と(d)の%～$までは特別な意味があります。次のセクションを参照してください。また、日本語は変数名に使えません。

(3)1文字目…変数の1文字目は、数字や特殊記号であってはいけません。

予約語

IFやFORなどのQuick BASICのステートメントは予約語ですので、プログラマが変数名としては用いることができません。

データの型と変数の型

数値型・文字型

コンピュータが処理するデータには、大きく分けて数値型のデータと文字型のデータがあります。

あるものの数や値段などは、数値型のデータです。逆に、人名や住所などは、文字型のデータです。

＄記号

文字型のデータを入れる器の名前(変数名)には、その最後にドル記号＄を付けます。

データの型と変数記号

数値型	A，B，…
文字型	A＄，B＄，…

接尾辞

変数の型を区別するために、変数名の最後に付ける＄などの特殊記号を接尾辞といいます。さらに、数値型のデータには、あまり大きくない整数(-32768～32767)を入れる整数型や、非常に大きな小数点付きの数値を入れる倍精度浮動小数点型などがあります。これらの区別も、変数名の最後に％や＆などの特殊記号(接尾辞)を付けて区別します。

接尾辞による変数の区分

接尾辞	データ型	範　囲
％	整数型	-32768～32767
＆	長い整数型	-2,147,483,648～2,147,483,647
！	単精度浮動小数点型	$-3.37\times10^{\pm38}$～$3.37\times10^{\pm38}$
＃	倍精度浮動小数点型	$-1.67\times10^{\pm308}$～$1.67\times10^{\pm308}$

接尾辞の付かない変数は、単精度浮動小数点型の数値型のデータとなります。特に天文学的な数値を扱わなければ、数値型については区分する必要がないでしょう。

このセクションでは、文字型データを入れる変数には＄記号が付くこと、何も接尾辞の付かない数値は単精度浮動小数点、という2点を記憶しておいてください。

第2章　コンソールとの入出力

キーボードとディスプレイを合わせてコンソールといいます。
第2章では、ディスプレイ(画面)への表示と、キーボードからのデータの入力について学習します。

第2章の概要

■画面のクリア	CLS
■データの表示	PRINT
■複数のデータの表示	PRINT…, …, …
	PRINT ; … ; …
■空行の表示	PRINT
■非改行表示	PRINT… ;
■表示様式	PRINT USING
■カーソルの位置	LOCATE
■文字の色	COLOR
■データの入力	INPUT

画面のクリア　CLS

CLS

画面をクリアするコマンドはCLSです。

CLSの基本書式

```
CLS
```

【例題19】　画面のクリア　CLS

画面をクリアするプログラムを作成してください。

解答例

```
CLS     '画面のクリア
```

解説　■カーソル

CLSによって画面はクリアされて、カーソルは画面の左上(ホームポジション)に位置します。

■コメント

プログラムには、コメント(注釈)を書くことができます。

プログラムの中で、'以降その行はコメントとみなされます。コメントはプログラムの見やすさの問題で、実行には関係ありません。

2つの画面

画面には、文字情報が表示されるテキスト画面と、絵が表示されるグラフィック画面があります。通常この2枚の画面は、スーパーインポーズ(重なった状態でそれぞれ独立に表示)されています。

CLSは、その両方を一度にクリアします。テキスト画面のみ、あるいはグラフィック画面のみをクリアする場合は、それぞれCLS 2，CLS 1 とします。

データの表示 PRINT

PRINT

データを表示するコマンド(命令)は、PRINT です。

PRINT の基本書式

```
PRINT データ
```

ここでデータには次のものを書くことができます。

PRINT コマンドのデータ

(1)数値や文字などの単体データ
(2)数値や文字などの入っている変数
(3)(1)や(2)からなる式

【例題20】 数値データの表示 PRINT

自分の年齢を表示するプログラムを作成してください。

解答例(1)

```
PRINT 22
```

解答例(2)

```
AGE = 22
PRINT AGE
```

解説 **■プログラムの可読性**

プログラムの見やすさをプログラムの可読性といいます。上の解答例(1)と(2)では、(2)の方が意味がとらえやすく可読性の高いものといえます。

【例題21】　演算結果の表示　PRINT

5年後の自分の年齢を表示するプログラムを作成してください。

解答例(1)

```
AGE = 22
PRINT AGE + 5
```

解答例(2)

```
AGE = 22
AFTER = 5
PRINT AGE + AFTER
```

解説　**■プログラムの保守性**

上のプログラムを変更して、7年後の年齢を求めるプログラムとする場合、解答例(1)と(2)ではどちらが修正しやすいですか。私は(2)のほうだと思います。プログラムの時からの修正のしやすさを、プログラムの保守性といいます。

【例題22】　文字データの表示　PRINT

自分の姓と名前を表示してください。

解答例(1)

```
PRINT "山田"
PRINT "太郎"
```

解答例(2)

```
SEI$ = "山田"
MEI$ = "太郎"
PRINT SEI$
PRINT MEI$
```

実行例

```
山田
太郎
```

解説　■引用符

PRINT 文のデータとして文字型データを与える場合は、引用符("")でその文字をくくってください。もし、引用符でくくらないとどうなるでしょうか。やってみてください。

```
PRINT  YAMADA
(変数 YAMADA の持つ値 0 が表示されるでしょう)
```

■変数の型

データを入れる器としての変数は、その中に入ってくるデータの型(数値か文字かなど)に合わせる必要があります。文字を入れる変数には＄記号をつけます。

(例)　SEI$, MEI$

複数のデータの表示

PRINTでは2つ以上のデータを1行に表示することができます。ピタッとくっつけて表示する場合(密着型)と、ある程度の間隔をあけて表示する場合(等間隔型)では、書式が異なります。

複数のデータの表示

```
密着型     PRINT データ1; データ2; …;…     (セミコロン)
等間隔型   PRINT データ1, データ2, …,…     (カンマ)
```

【例題23】 複数データの表示(文字密着型)

変数SEI$には山田、変数MEI$には太郎の文字列を入れて、この2つの変数を使って

山田 太郎

と表示してください。

解答例

```
SEI$ = "山田"
MEI$ = "太郎"
PRINT SEI$ ; " "; MEI$
```

実行例

```
山田 太郎
```

解説

■" "

姓と名との間をあけるためには、" "で表現するスペースが必要です。

■文字列のたし算

文字列のたし算を行なって、次のようにPRINT文を使っても同じ答が得られます。試してください。

文字列のたし算による表示

```
PRINT SEI$ + " " + MEI$
```

【例題24】　複数データの表示(数値密着型)

彼の年は18(HE=18)、彼女の年は16(SHE=16)として2人の年齢を密着して表示してください。

解答例

```
HE  =  18
SHE  =  16
PRINT  HE; SHE
```

実行例

```
18␣␣16
```

解説　■符号分

残念ながら、2人の年齢はピッタリとはくっつきませんでした。これは、数値データには、マイナス符号をつけるために頭に一文字分の余裕がとられているからです。

そしてさらに、PRINTは数値データ出力後に1スペースあけます。したがって、18と16の間は2スペース分あきます。

【例題25】　複数データの表示(文字等間隔型)

次の4つの文字列を等間隔に表示してください。
ABC, XYZ, ABC, PQR

解答例

```
PRINT  "ABC","XYZ","ABC","PQR"
```

実行例

```
ABC           XYZ           ABC           PQR
↑             ↑             ↑             ↑
1桁目         15桁目        29桁目        43桁目
```

解説　■14桁

PRINT文のデータをカンマで区切った場合は、14桁ずつの等間隔表示になります。

【例題26】　複数データの表示(数値等間隔型)

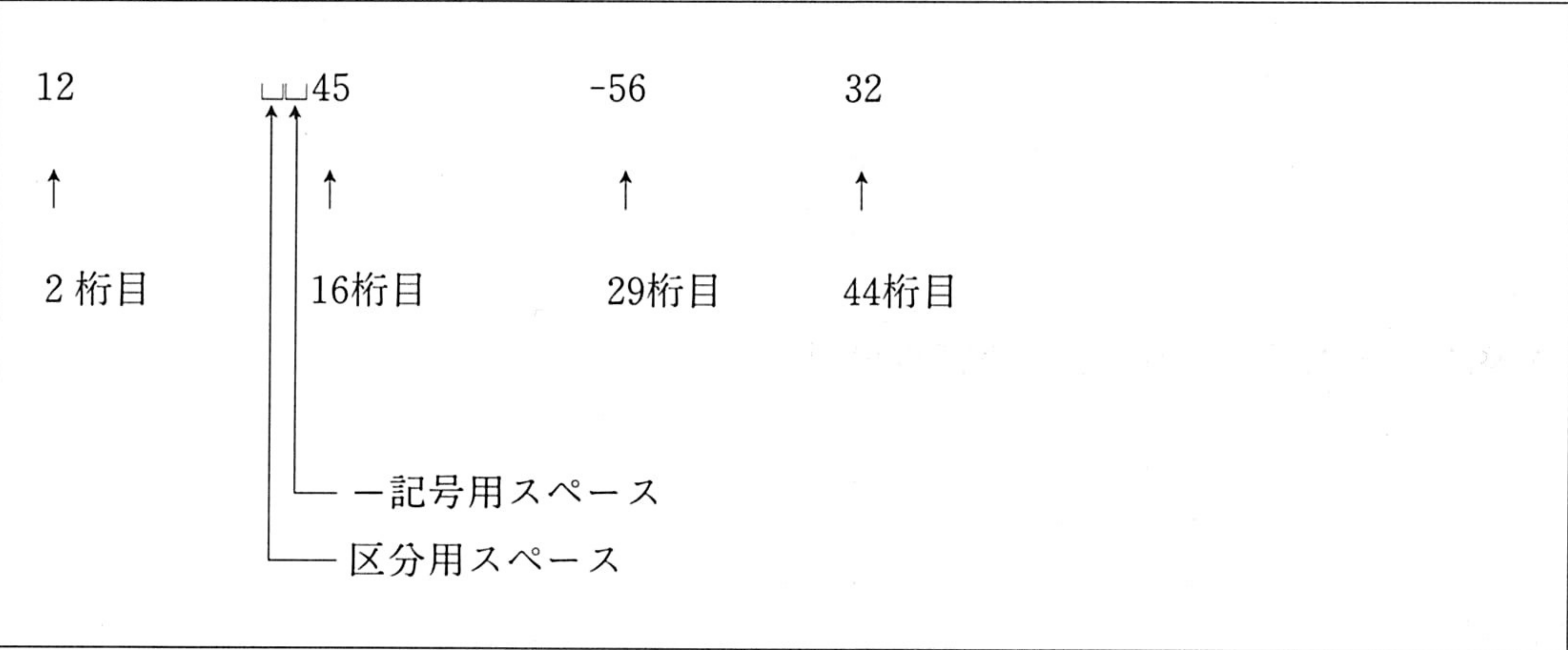
次の4つの数字を等間隔に表示してください。
　12, 45, -56, 32
前ページの **【例題25】** との表示の差に気を付けてください。

解答例

```
PRINT  12, 45, −56, 32
```

実行例

```
12            ␣␣45           -56          32
↑               ↑              ↑           ↑
2桁目          16桁目          29桁目       44桁目

              ├─ 記号用スペース
              └─ 区分用スペース
```

解説　**■符号分**

数値データでは符号用に1つ、出力後に1つのスペースが入ります。

■表示様式

数値データについて桁などをそろえて表示する場合は、PRINT　USING を使うか、あるいは文字列に変換してから表示するとうまくいきます。

空行・非改行表示

空行

データを与えないでPRINTだけを書くと、空行が表示されます。

【例題27】　空行の表示

SEI$ = "山田",MEI$ = "太郎"のとき、次のように空行を入れて表示してください
山田

太郎

解答例

```
SEI$ = "山田"
MEI$ = "太郎"
PRINT SEI$
PRINT
PRINT MEI$
```

実行例

```
山田

太郎
```

非改行表示

PRINTで表示するデータの最後にセミコロンをつけておくと、続くPRINTによる表示は、前のデータに続けて表示(非改行表示)されます。これは、PRINTのデータが多くて1行に書ききれないときに使うと便利です。

【例題28】　非改行表示

SEI$ = "山田",MEI$ = "太郎"のとき、PRINTを2回使って
　山田 太郎
と表示させてください。

解答例

```
SEI$ = "山田"
MEI$ = "太郎"
PRINT "SEI$";" ";
PRINT "MEI$"
```

実行例

```
山田 太郎
```

音をPRINTする？

PRINT CHR$(7)

としてみてください。ピッと音がするでしょう。PRINTで音が出せるなんて少し変ですが…。CHR$(7)は、ブザーを意味しています。

ブザーを鳴らすステートメントとしてBEEPがありますので、PRINT CHR$(7)は使うことがないでしょうが、PRINTの珍しい使い方として紹介しました。

表示様式 PRINT USING

数値型のデータでは、3桁ごとのカンマや、頭に¥などを付けることが実務上よく使われます。PRINT USINGという構文を使うと、表示する様式を指定することができます。以下の例で、♯は任意の1数字に該当します。必要な桁数♯を書きます。

3桁カンマ

数値データに3桁カンマを入れるときは、次のようにします。

```
PRINT USING "♯♯,♯♯♯";データ
```

¥記号

数値データの頭に¥記号を表示する場合は、次のようにします。

```
PRINT USING "¥¥♯♯♯♯♯";データ
```

¥記号と3桁カンマを入れる場合は、次のようにします。

```
PRINT USING "¥¥♯♯,♯♯♯";データ
```

【例題29】 表示様式 PRINT USING

値31198を3桁カンマ付きで表示してください。また、値123に¥記号を付けて表示してください。

解答例

```
PRINT USING "♯♯,♯♯♯";31198
PRINT USING "¥¥♯♯♯";123
```

カーソルの位置 LOCATE

テキスト画面座標

次に、データの表示や入力を画面上の任意の位置で行なうことを学習しましょう。

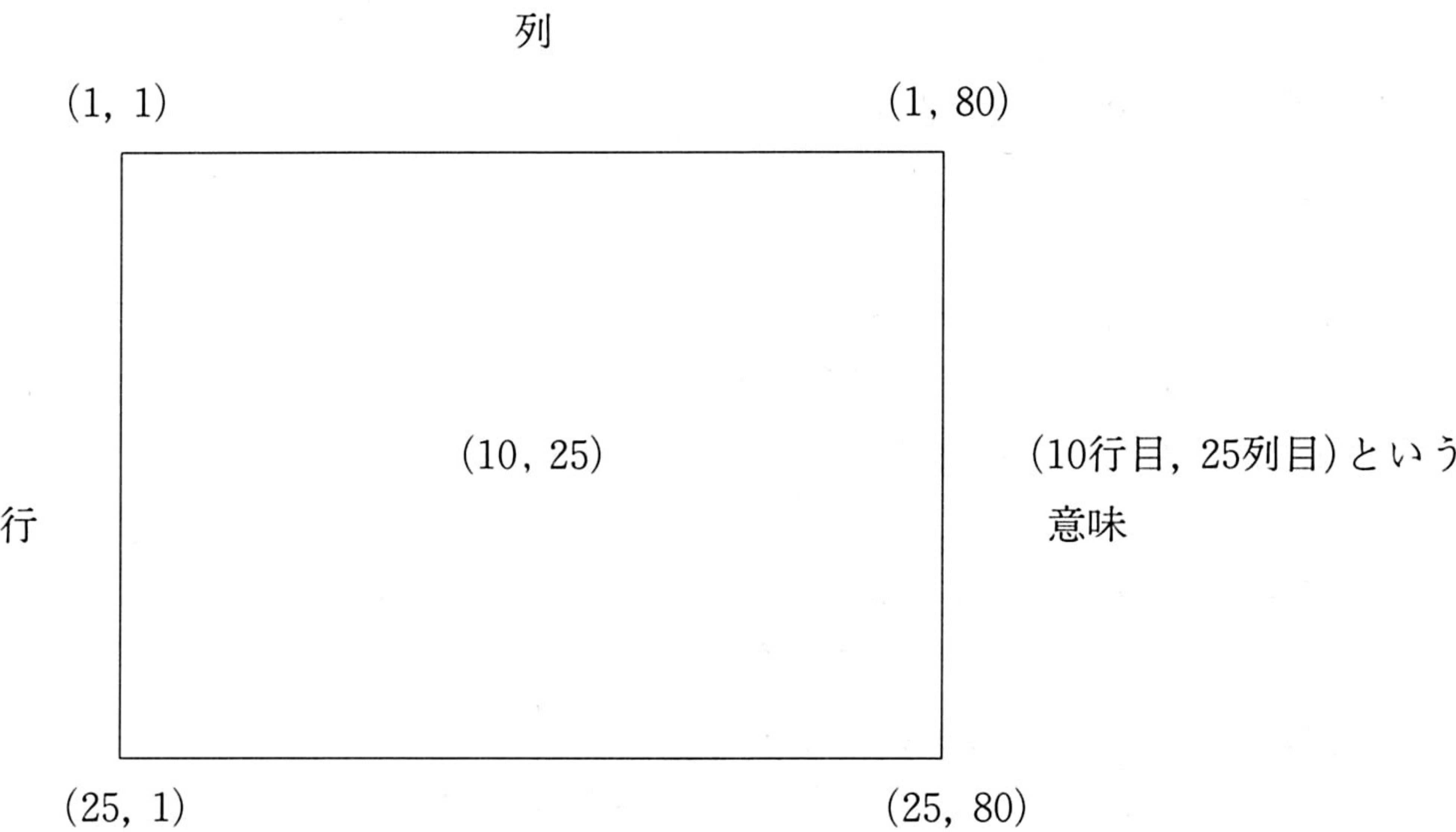
画面は横に80列、縦に25行あります。カーソルの位置は、画面左上を(1, 1)とする座標の考え方で示します。座標表示は(行, 列)の順です。

カーソルの位置

列

(1, 1)　(1, 80)

行　(10, 25)　(10行目, 25列目)という意味

(25, 1)　(25, 80)

LOCATE

カーソルの位置は、LOCATEコマンドによって設定できます。

LOCATEの基本書式

```
LOCATE 行, 列
```

【例題30】 カーソルの位置 LOCATE

画面のほぼ中央から　こんにちは　と表示してください。

解答例

```
CLS
LOCATE 12, 40
PRINT "こんにちは"
```

実行例

```
こんにちは
```

文字の色 COLOR

色

PRINT 文による文字の表示は、通常黒地のスクリーンに白の文字で書かれます。Quick BASIC では、文字の色とバックとなる画面全体の色を指定できます。文字の色と画面の色を決めるステートメントは、COLOR です。PC-9801 シリーズと AX パソコンでは、その意味することが違っています。

COLOR

COLOR の基本書式(PC-9801 シリーズ)

```
COLOR 文字の色，画面の色
```

COLOR の基本書式(AX パソコン)

```
COLOR 文字の色，文字背景の色
```

ここで文字の色や画面の色は、使用するコンピュータや、その初期設定などで異なります。

カラーコード(PC-9801 シリーズ)

文字の色	画面の色
0(黒)	0
1(青)	1
2(緑)	2
3(水)	3
4(赤)	4　(同左)
5(紫)	5
6(黄)	6
7(白)	7
8～15	8～15
(0～7の反転)	(0～7の暗い色)
16～31	(ただし、80286
(0～15の点滅)	10MHz＋アナログディ
	スプレイに限る。)

カラーコード(AX パソコン)

文字の色	文字背景の色
0(黒)	0
1(青)	1
2(緑)	2
3(水)	3
4(赤)	4　(同左)
5(紫)	5
6(黄)	6
7(白)	7
8～15	
(0～7の明るい色)	
16～31	
(0～15の点滅)	

(例)

```
COLOR 4,7
PRINT "XXX"
```

PC-9801 シリーズ　画面全体が7(白)

XXX←―― 文字が4(黄)

AX パソコン　文字の後ろが7(白)

[XXX]←―文字が4(黄)

【例題31】 文字の色 COLOR

黄色い字で自分の名前を書いてください。

解答例

```
COLOR 6
PRINT "山田 太郎"
```

解説 ■元に戻す

特別色を使ったあとは、必ず元に戻しておいてください。

COLOR 7, 0

が黒いスクリーンに白の文字です。

【例題32】 反転 COLOR

画面の最下段中央(24行目, 40桁)に、黄色の反転文字で注意! と書いてください。

解答例

PC-9801

```
LOCATE 24,40
COLOR 14
PRINT "注意!"
```

AX パソコン

```
LOCATE 24, 40
COLOR 0, 6
PRINT "注意!"
```

解説 ■AX パソコン

AX パソコンでは、文字の色と文字の背景色を別々に指定することになります。黄色の反転ですから、文字の背景色を黄色、文字の色を黒にします。

■PC-9801

PC-9801 では、文字の色として直接黄色の反転がカラー番号 6+8=14 で指定できます。

データの入力 INPUT

INPUT

キーボードからデータを入力するコマンドは、INPUT です。

INPUT の基本書式

```
INPUT　プロンプト文，変数
```

プロンプト文

プロンプト文は、入力を案内するための文字列です。これは、あってもなくてもかまいません。プロンプト文と変数を区切るのは通常カンマ，を使いますが、セミコロン；を使うこともできます。プロンプト文と変数をセミコロンで区切ると、実行時にプロンプト文の次にクエスチョンマーク？が表示されます。

変数の型

データを入れる器としての変数は、その中に入るデータの型（数値か文字かなど）に合わせなくてはなりません。くどいようですが、文字を入れる変数(文字型変数)には、その最後に＄をつけてください。

【例題33】　データの入力(数値)　INPUT　A

> キーボードから2つの数値を入力して、その差を表示するプログラムを作成してください。

解答例(1)

```
INPUT　A
INPUT　B
PRINT　A　－　B
```

解答例(2)

```
INPUT　"第1の数値",A
INPUT　"第2の数値",B
PRINT　"その差";A　－　B
```

解答例(3)

```
INPUT　"第1の数値";A
INPUT　"第2の数値";B
PRINT　"その差";A　－　B
```

実行例(解答例(1))

```
?6 ⏎
?5 ⏎
1
```

実行例(解答例(2))

```
第1の数値　6 ⏎
第2の数値　5 ⏎
その差　1
```

実行列(解答例(3))

```
第1の数値　?　6 ⏎
第2の数値　?　5 ⏎
その差　1
```

解説　■?マーク

プロンプト文を付けない解答例(1)と、プロンプト文と変数を;で区切った解答例(3)では、入力に際して?マークが表示されます。

【例題34】　データの入力(文字)　INPUT　A$

キーボードから姓と名を入力して、その2つを合成して(足し加えて)フルネームで表示してください。

解答例(1)

```
INPUT ”姓”,SEI$
INPUT ”名”,MEI$
PRINT SEI$ + ” ” + MEI$
```

解答例(2)

```
INPUT ”姓”,SEI$
INPUT ”名”,MEI$
PRINT SEI$;” ”;MEI$
```

実行例

```
姓 山田 ⏎
名 太郎 ⏎
山田 太郎
```

第3章　分岐と繰り返し

本章では、プログラムの実行順を変えてみましょう。まず最初に、プログラムの流れを変えるきっかけとなる条件とその評価について考え、次に分岐や繰り返しといった本題に入ります。

第3章の概要

■条件式とその評価

条件式は、関係演算子、論理演算子を含む式です。条件式は、その条件が満たされているときは0以外の値、条件が満たされていないときは0になります。

■条件判断　IF ～ END IF

条件によってプログラムの流れを変えるステートメントはIFです。IFには、派生的な書式がいくつかあります。

■場合分け　SELECT CASE ～ END SELECT

IF文では、複雑になる多方向への場合分けは、SELECT CASE文によって処理します。

■繰り返し（1）　FOR ～ NEXT

FORステートメントは、一定の回数同じ動作を繰り返すことに適しています。

■繰り返し（2）　WHILE ～ WEND

WHILE ～ WENDの構文は、キッチリとした構造化プログラムを書くのに適しています。

■繰り返し（3）　DO ～ LOOP

DO ～ LOOPは、もっとも応用性の高いループ構造（繰り返し）を表現できます。

■プログラムの停止と終了　STOP, END

条件式とその評価

「AならばBをする」というときのAを条件といいます。条件は式で表現されます。

等しい、小さいなど、2つの数値間の関係は関係演算子で表現されます。

関係演算子

関係演算子

aはbに等しい	a＝b
aはbに等しくない	a＜＞b
aはbより大きい	a＞b
aはbより小さくない	a＞＝b
aはbより小さい	a＜b
aはbより大きくない	a＜＝b

これら関係演算子で結ばれた式の値は、その条件が満たされているとき(真のとき)は、0以外の整数になります。満たされていないとき(偽のとき)は、値が0になります。

(例)　5＝10　→　値0、5＜＞10　→　0以外の整数

条件が複数個あるときの各条件間の関係は論理演算子で決めます。

論理演算子

論理演算子

条件Aも条件Bも	A　AND　B
条件Aか条件B	A　OR　B

論理演算子のひとつに否定演算子があります。

否定

論理演算子(否定)

Aでないとき	NOT　A

論理演算子で結ばれた条件式も、それらが総合的に満たされれば0以外の整数、満たされなければ0に評価されます。

条件判断　IF ～ END IF

条件によって実行させる命令が異なる場合は、IF を使います。IF にはいろいろな派生書式がありますが、基本書式は次のとおりです。

IF の基本書式

```
IF　条件　THEN
　　文1　　条件が満たされているとき実行する文
ELSE
　　文2　　条件が満たされていないとき実行する文(必要なければ省略可)
END　IF
```

ELSEIF

ELSE の次に、IF が続く ELSEIF というキーワードが使えます。ELSEIF は対応する END　IF を必要としません。

【例題35】　条件判断　IF　～　END　IF

2つの数値を入力して、大きい方を表示するプログラムを作成してください。

解答例 (1)

```
INPUT A
INPUT B
IF A > B THEN
    PRINT A
ELSE
    PRINT B
END IF
```

解答例 (2)

```
INPUT A
INPUT B
IF B > A THEN
   A = B
END IF
PRINT A
```

実行例

```
?12 ⏎
?45 ⏎
 45
```

解説　　**■字下げ**

特別な場合に限って実行される命令文は、4文字(程度)字下げして書くと読みやすくなります。

【例題36】　条件判断　IF ～ ELSE ～ END IF

入力された数値が正の数ならプラス、負の数ならマイナス、0ならゼロと表示するプログラムを作成してください。

解答例

```
INPUT A
IF A > 0 THEN
     PRINT "プラス"
ELSE
     IF A < 0 THEN
          PRINT "マイナス"
     ELSE
          PRINT "ゼロ"
     END IF
END IF
```

実行例

```
?−12 ⏎
マイナス
```

IFは他のIF～ELSEの中に入れ子することができます。入れ子にしたIFについても、キチンとEND IFを対応させてください。

【例題37】 条件判断　ELSEIF

入力された数値が正の数ならプラス、負の数ならマイナス、0ならゼロと表示するプログラムを作成してください。
ただし、ELSEIFを使ってください。

解答例

```
INPUT  A
IF  A  >  0  THEN
      PRINT  "プラス"
ELSEIF  A  <  0  THEN
      PRINT  "マイナス"
ELSE
      PRINT  "ゼロ"
END  IF
```

解説　■ELSEIF

ELSEIFは、対応するEND IFを必要としません。ELSEIFを使うか、IFを入れ子にして使うかは好みの問題ですが、慣れるまではIFを入れ子にしてEND IFで閉じて完全な構造化プログラミングをするのがよいと思います。
ELSEIFを使わない解答例を下に示します。

```
INPUT  A
IF  A  >  0  THEN
      PRINT  "プラス"
ELSE
      IF  A  <  0  THEN
            PRINT  "マイナス"
      END  IF
      PRINT  "ゼロ"
END  IF
```

場合分け　SELECT CASE ～ END SELECT

条件による場合分けも、その数が多くなるとIF文で書くのはすっきりしなくなります。多くの場合分けに効率よく対応するステートメントは、SELECT CASEです。

SELECT CASEの基本書式

```
SELECT CASE 基準
      CASE 条件1
              文1
      CASE 条件2
              文2
               ⋮
      CASE ELSE
              文n
END SELECT
```

条件　　条件は、次のように表現します。

(1)値の一致　　SELECT CASE A　　A＝12のとき
　　CASE 12

(2)値の範囲の一致　　SELECT CASE A　　Aが１～12のとき
　　CASE 1 TO 12

(3)大小関係　　SELECT CASE A　　Aが13より大きいとき
　　CASE IS ＞ 13

複数の条件

条件がいくつかある場合は、その条件をカンマで区切って表現します。

(例)　SELECT CASE A　　Aが１，２，５いずれかのとき
　　CASE 1, 2, 5

【例題38】 SELECT CASE <数値による分岐> ～ END SELECT

ひと桁の数値(1～5)を入力し、その英語(ONE～FIVE)を表示するプログラムを作成してください。規定外の入力には、ブザーを鳴らしてください。ブザーを鳴らすステートメントはBEEPです。

解答例

```
INPUT A
SELECT CASE A
    CASE 1
        PRINT "ONE"
    CASE 2
        PRINT "TWO"
    CASE 3
        PRINT "THREE"
    CASE 4
        PRINT "FOUR"
    CASE 5
        PRINT "FIVE"
    CASE ELSE
        BEEP
END SELECT
```

解説　■CASE ELSE

場合分けした条件のいずれにも一致しない場合は、CASE ELSEの次の文(この場合はBEEP)が実行されます。

【例題39】 SELECT CASE ＜文字列による分岐＞ ～ END SELECT

前ページの逆、すなわちONEと入れると1、TWOと入れると2を表示するプログラムを作成してください。

解答例

```
INPUT A$
SELECT CASE A$
    CASE "ONE"
        PRINT "1"
    CASE "TWO"
        PRINT "2"
    CASE "THREE"
        PRINT "3"
    CASE "FOUR"
        PRINT "4"
    CASE "FIVE"
        PRINT "5"
    CASE ELSE
        BEEP
END SELECT
```

解説　**■文字列**

SELECT CASEでは、数値的な条件だけでなく、文字列との一致もその条件とすることができます。

【例題40】 SELECT CASE ＜複数の条件＞ ～ END SELECT

0から9までの数値を入力し、奇数ならば奇数、偶数であれば偶数と表示するプログラムを作成してください。

解答例

```
INPUT A
SELECT CASE A
    CASE 0, 2, 4, 8
        PRINT "偶数"
    CASE 1, 3, 5, 7, 9
        PRINT "奇数"
END SELECT
```

解説　**■複数の条件**

CASE に続く条件をカンマで区切ると、そのいずれかの意味になります。

【例題41】 SELECT CASE IS <大小関係> ～ END SELECT

入力された数値が10より大きいときは大きい、20より大きいときはかなり大きい、30より大きいときは大きすぎると表示してください。

解答例

```
INPUT A
SELECT IS > 30
    CASE IS > 30
        PRINT "大きすぎる"
    CASE IS > 20
        PRINT "かなり大きい"
    CASE IS > 10
        PRINT "大きい"
END SELECT
```

解説　**■上から順に**

CASE 文ではひとつ該当するものがあれば、それ以降にもし該当するものがあっても無視されます。このことは、上の解答例を10，20，30の順に判定するように書き直すと分かります。

【例題42】 SELECT CASE <範囲の指定> TO ～ END SELECT

入力された数値が1から4、あるいは10～40のときは"OK"、5から8あるいは50～80のときは"NG"を表示するプログラムを作成してください。

解答例

```
INPUT A
SELECT CASE A
     CASE 1 TO 4, 10 TO 40
          PRINT "OK"
     CASE 5 TO 8, 50 TO 80
          PRINT "NG"
END SELECT
```

解説 ■**範囲の指定**

範囲の指定には、TOを用います。

繰り返し(1) FOR ～ NEXT

同じことを何回か限られた回数繰り返す場合は、FORを使います。FOR文の基本書式は次のとおりです。

FORの基本書式

```
FOR カウント変数 = 開始 TO 終了 [STEP 増分]
        文
NEXT カウント変数

(増分が1のとき[STEP 増分]は省略できます)
```

脱出　正規の終了回数を待たずにFORループの中から抜け出るには、EXIT FORを使います。

【例題43】 繰り返し FOR ～ NEXT

自分の名前を10回繰り返して表示するプログラムを作成してください。

解答例

```
FOR I = 1 TO 10
    PRINT "山田 太郎"
NEXT I
```

解説　**■カウント変数**

FOR文で回数を数えるための変数をカウント変数(カウンタ)といいます。

【例題44】 変数の初期化　FOR ～ NEXT

１から10までの合計を求めるプログラムを作成してください。

解答例

```
SUM  =  0
FOR  I  =  1  TO  10
     SUM  =  SUM  +  I
NEXT  I
PRINT  SUM
```

解説　■変数の初期化

合計を蓄える変数SUMは、プログラムの先頭で０にクリアする必要があります。これを変数の初期化といいます。

【例題45】 増分　STEP

１から50までの奇数の合計を求めるプログラムを作成してください。

解答例

```
SUM  =  0
FOR  I  =  1  TO  50  STEP  2
     SUM  =  SUM  +  I
NEXT  I
PRINT  SUM
```

解説　■STEP 2

奇数は１から始まって２飛びですから、STEP　２とすることによって1, 3, 5, 7…の合計を求めることができます。

【例題46】　脱出　EXIT　FOR

10回データを入力して、その入力のたびにそれが10以下ならば表示するプログラムを作成してください。もし、0が入力されれば、10回に満たなくても強制的に終了してください。

解答例

```
FOR I = 1 TO 10
    INPUT A
    IF A = 0 THEN
        EXIT FOR
    END IF
    IF A <= 10 THEN
        PRINT A
    END IF
NEXT I
```

解説　■脱出　EXIT　FOR

一定の条件でFORによる繰り返しを強制的に終了する場合は、EXIT　FORを使用してください。

少し待つ

少しの時間プログラムの進行を止めたいときは、何も実行しないFOR文を使います。

```
FOR I = 1 TO 10000
NEXT I
```

コンピュータの性能によってどれくらいの時間止まるかは異なりますが、上の例では2～3秒でしょう。TO　nのnを変更して時間を調整してください。

【例題47】 FORの入れ子

> 自分の名前を2回ずつ5回書きなさい。ただし次のように空白行を入れてください。
>
> 山田太郎
> 山田太郎
>
> 山田太郎
> 山田太郎
>
> ⋮

解答例

```
FOR I = 1 TO 5
    FOR J = 1 TO 2
        PRINT "山田 太郎"
    NEXT J
    PRINT
NEXT I
```

解説

■FORの入れ子

FOR文の中でFORを回すことができます。これをFORの入れ子といいます。また、コンピュータにとっては意味のないことですが、プログラム記述上は字下げをして読みやすくする工夫が必要です。

■空白行

データなしでPRINTとだけ書くと、空白行が表示されます。

繰り返し (2) WHILE ～ WEND

条件が満たされている間、一定の文を繰り返し実行するには、WHILEを使います。WHILEの基本書式は次のとおりです。

WHILEの基本書式

```
WHILE 条件
    文
WEND
```

【例題48】 繰り返し　WHILE ～ WEND

1から10までの合計を求めるプログラムを作成してください。

解答例

```
J = 1
SUM = 0
WHILE J < 11
    SUM = SUM + J
    J = J + 1
WEND
PRINT SUM
```

解説　**■インクリメント　カウント変数の増加**

WHILE～WENDには、カウント変数を自動的に更新する機能はありません。カウンタ動作をさせるには、プログラムの中で明示的に J = J + … でインクリメント（増分の足し加え）してください。

【例題49】　WHILEの増分

１から50までの奇数の合計を求めるプログラムを作成してください。

解答例

```
I = 1
SUM = 0
WHILE I < 51
      SUM = SUM + I
       I = I + 2
WEND
PRINT SUM
```

解説　■増分

WHILE～WEND機構では、増分もプログラマの責任で調整してください。

【例題50】　前判断

キーボードからＱが入力されるまで、入力された文字を表示するプログラムを作成してください。

解答例

```
CLS
A$ = "A"
WHILE A$ <> "Q"
      INPUT A$
      PRINT A$
WEND
```

実行例

```
?B
B
?Q
Q

何かキーを押してください
```

解説　■前判断

WHILE～WENDは、実行文の実行以前に条件判断をしますから、上の例のように、入力以前にA$ ＝ "A"などとして、少なくとも１回はループの中が実行されるようにする工夫が必要です。

【例題51】　無限ループ

自分の名前をいつまでも表示するプログラムを作成してください。

解答例

```
TRUE = 1
WHILE TRUE
      PRINT "山田 太郎"
WEND
```

解説　**■TRUE = 1**

WHILE は、それに続く条件の真偽を次のように判定します。

条件式が0でない値を返しているとき　→　真
条件式が0を返しているとき　　　　　→　偽

したがって、WHILE TRUE はいつまでもその条件が偽とならない無限ループとなります。

■STOP

上のプログラムを実行すると、STOP キーの押し下げ以外に止める手段がありません。

■脱出

WHILE～WEND のループから脱出するためのステートメントは用意されていません。必要ならば、プログラム全体を終了させる END 文を使ってください。

(例)　100回自分の名前を書いて終了

```
N = 1
TRUE = 1
WHILE TRUE
      PRINT "山田 太郎"
      N = N + 1
      IF N > 100 THEN
            END
      END IF
WEND
```

繰り返し (3)　DO ～ LOOP

プログラムの繰り返しを設定するステートメントには、もうひとつDO ～ LOOPがあります。

DO ～ LOOPには次の４つの書式があります。

DO ～ LOOPの４書式

```
(1)DO  WHILE  条件
          文
   LOOP

(2)DO  UNTIL  条件
          文
   LOOP

(3)DO
          文
   LOOP  WHILE  条件

(4)DO
          文
   LOOP  UNTIL  条件
```

上の書式で(1)と(2)は文の実行以前に条件判定をする前判定、(3)と(4)は文の実行後に判定する後判定になります。

【例題52】　繰り返し　DO ～ LOOP

１から10までの合計を求めるプログラムを作成してください。

解答例

```
(1)SUM = 0
 I = 1
 DO WHILE I < 11
     SUM = SUM + I
     I = I + 1
 LOOP
 PRINT SUM
(2)SUM = 0
 I = 1
 DO UNTIL I = 11
     SUM = SUM + I
     I = I + 1
 LOOP
 PRINT SUM
(3)SUM = 0
 I = 1
 DO
     SUM = SUM + I
     I = I + 1
 LOOP WHILE I < 11
 PRINT SUM
(4)SUM = 0
 I = 1
 DO
     SUM = SUM + I
     I = I + 1
 LOOP UNTIL I = 11
 PRINT SUM
```

解説

■WHILEとUNTIL

WHILEは〜の間ということを意味します。UNTILは〜までということを意味します。UNTILは、その〜までに達したときにその内容を実行せずにループを抜けます。

【例題53】 無限ループと脱出 EXIT DO

999が入力されるまで、入力された数値の合計(最後の999は合計に含めない)を求めるプログラムを作成してください。

解答例

```
(1)SUM = 0
  TRUE = 1
  DO WHILE TRUE
        INPUT A
        IF A = 999 THEN
             EXIT DO
        END IF
        SUM = SUM + A
  LOOP
  PRINT SUM
(2)SUM = 0
  FALSE = 0
  DO UNTIL FALSE
        INPUT A
        IF A = 999 THEN
             EXIT DO
        END IF
        SUM = SUM + A
  LOOP
  PRINT SUM
(3)SUM = 0
  TRUE = 1
  DO
        INPUT A
        IF A = 999 THEN
             EXIT DO
```

```
        END  IF
        SUM  =  SUM  +  A
  LOOP  WHILE  TRUE
  PRINT  SUM
(4)SUM  =  0
  FALSE  =  0
  DO
        INPUT  A
        IF  A  =  999  THEN
            EXIT  DO
        END  IF
        SUM  =  SUM  +  A
  LOOP  UNTIL  FALSE
  PRINT  SUM
```

解説　　**■脱出**

DO～LOOP のループから脱出する場合は、EXIT　DO を使用してください。

プログラムの停止と終了　STOP, END

STOP

プログラムをプログラムの中から停止させることができます。プログラムの進行に合わせて、一時的に止めたいところにSTOPを書いてください。STOP以前を実行してエディットモードに戻ります。STOP以降を続けて実施するには、SHIFT＋f·5ではなく、単にf·5を押してください。

END

プログラムを終了するステートメントは、ENDです。

【例題54】　プログラムの停止と終了　STOP, END

WHILE～WENDを用いて数値を無限回入力するループを作成し、0が入力されたら停止、999が入力されたら終了させるプログラムを作成してください。

解答例

```
TRUE = 1
WHILE TRUE
    INPUT A
    IF A = 0 THEN
        STOP
    END IF
    IF A = 999 THEN
        END
    END IF
WEND
```

解説　■デバッグ

プログラムが自分の予想したとおりに働かない場合など、STOPを適当な箇所に書いてプログラムを部分的にデバッグしていくことができます。

第4章　グラフィックI

Quick BASICが他の言語に比べて優れている点のひとつに、その言語仕様の中に豊富なグラフィックコマンドを持っていることが挙げられます。

第4章では、グラフィックの基本的なコマンドを学習します。複雑な模様塗りやアニメーションの基礎については、第3部で解説します。

第4章の概要

■グラフィック画面

グラフィック画面は、

PC-9801では640×400ドット

AXパソコンでは640×480ドット

が基本です。

■カラー

カラーは、基本的には8色です。16色あるいは64色を使うこともできます。

■点を描く・消す	PSET, PRESET
■線を描く	LINE
■四角を描く	LINE　Bオプション, BFオプション
■円を描く	CIRCLE
■円弧と扇型	CIRCLE　角度オプション
■楕円	CIRCLE　比率オプション
■色を塗る	PAINT
■色を変える	PALETTE
■色を調べる	POINT
■連続描画	DRAW
■グラフィック範囲の限定	VIEW

＊AXパソコン

AXパソコンでは、プログラムの先頭にSCREEN　0という一行を入れて、グラフィック画面の設定をしてください。

グラフィック画面

グラフィック画面の座標は、テキスト画面(文字を書く画面)とは異なります。
グラフィック画面の座標は、次のとおりです。

(0,0) (639, 0)

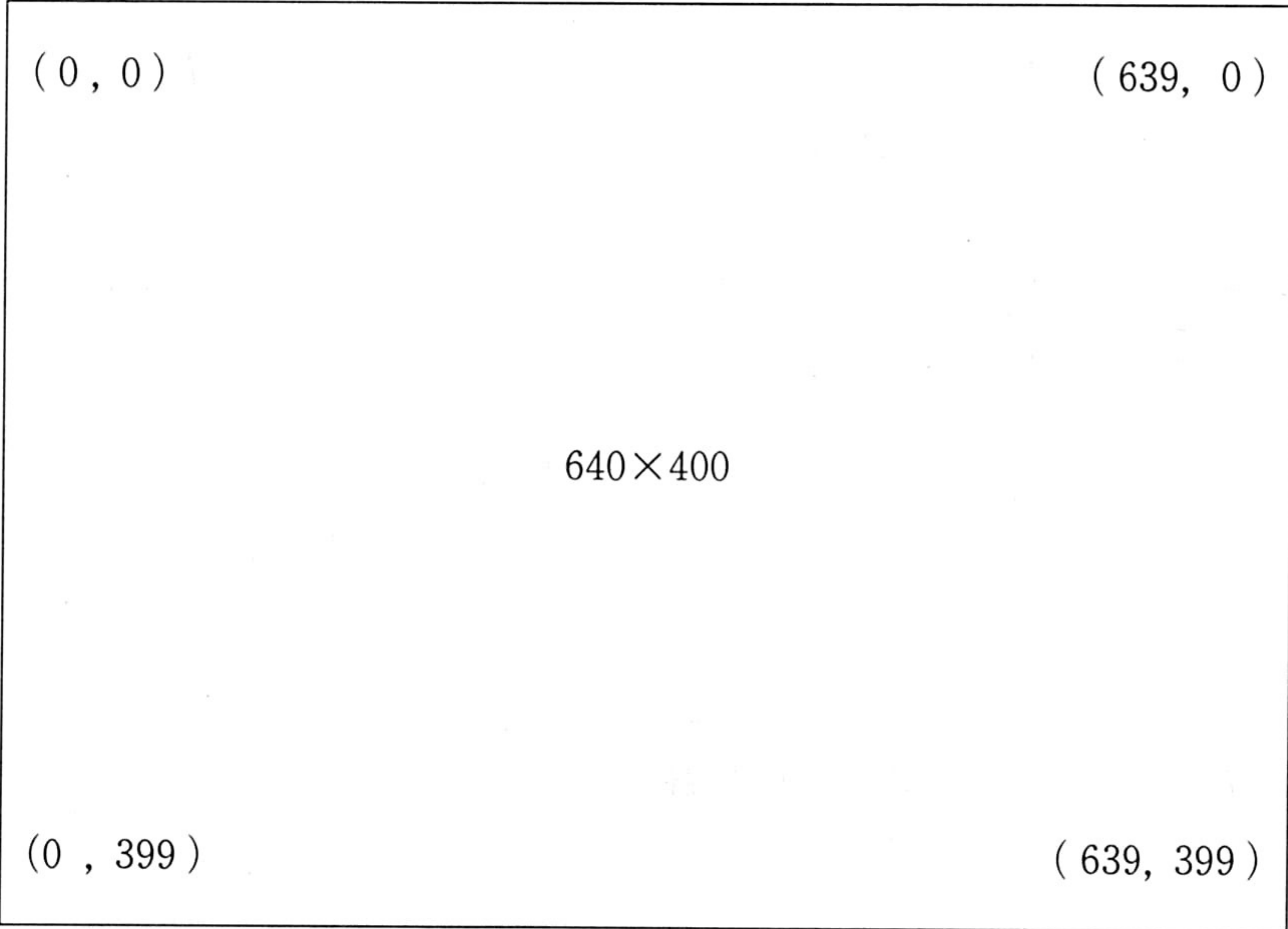
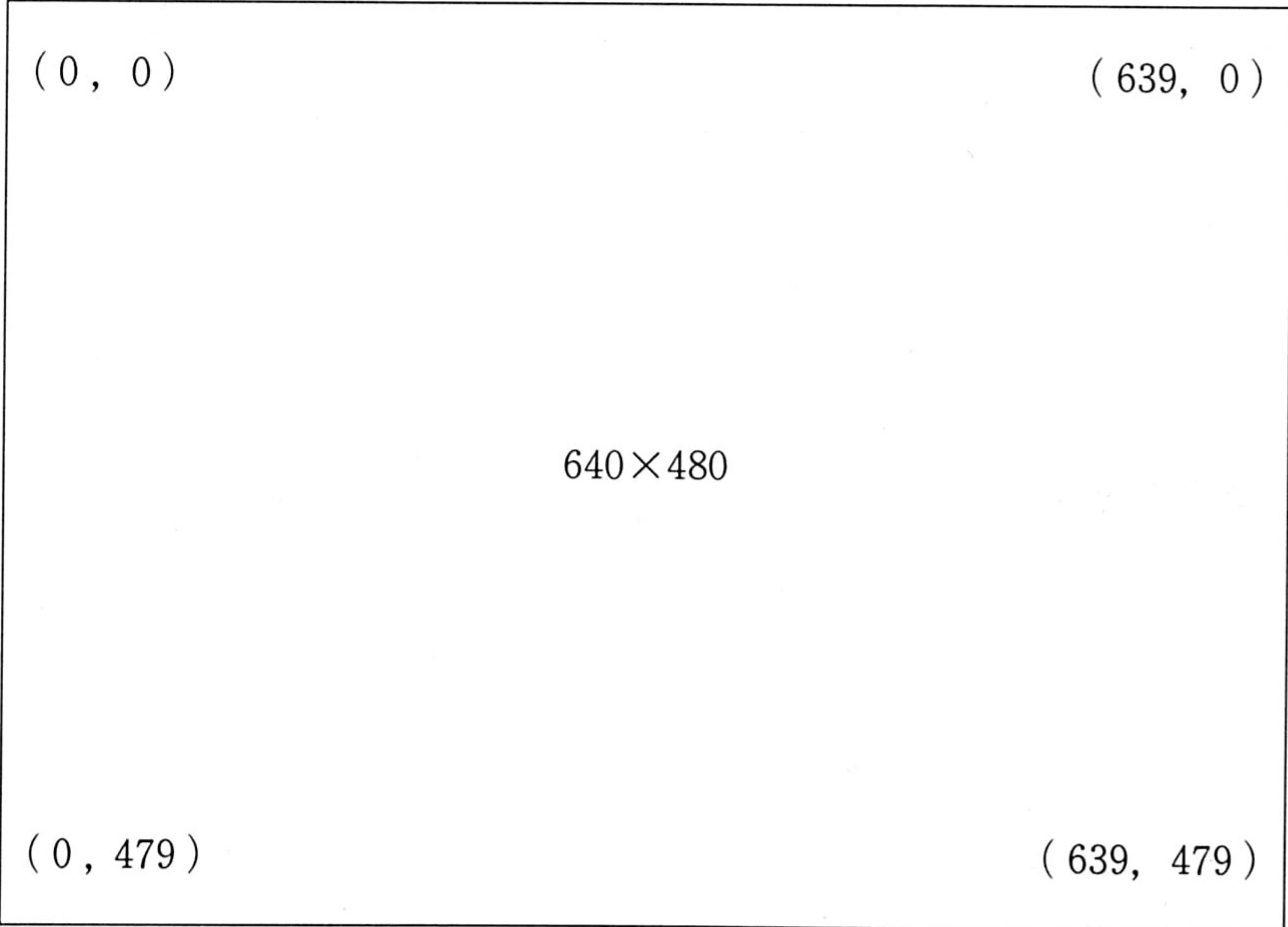
640×400

(0 , 399) (639, 399)

PC-9801

(0, 0) (639, 0)

640×480

(0,479) (639, 479)

AX パソコン

カラー

グラフィック画面で使える基本的なカラーは次の８色または16色です。左にカラー番号が書いてあります。これは、次から学習する点や線を描くときのカラーを指定する番号になります。

カラー番号（PC-9801）

番号	色
0	黒
1	青
2	緑
3	水
4	赤
5	紫
6	黄
7	白
8 〜15	0 〜 7 の暗 い 色

カラー番号（AX パソコン）

番号	色
0	黒
1	青
2	緑
3	水
4	赤
5	紫
6	黄
7	白
8 〜15	0 〜 7 の明るい色

カラー

PC-9801 では、CPU が 80286 で 10MHz動作＋アナログディスプレイ の場合に限って、８色（０〜７）だけではなく16色以上の色（０〜15）が使えます。

本書では、基本色８色によるプログラミング例を示すにとどめます。

AX パソコン

AX パソコンでは、SCREEN　0 と指定することで16色が使えます。以下の解答例では、SCREEN　0 を最初に書いてください。

グラフィック画面のクリア

グラフィック画面をクリアするステートメントはCLS です。テキスト文だけを残しておきたい場合はCLS　1、逆にグラフィック画面を残しておきたい場合はCLS　2 としてください。

点を描く PSET

グラフィック画面に点を描くコマンドは、PSETです。PSETの基本書式は、次のとおりです。

PSETの基本書式

```
PSET (X座標, Y座標), カラーコード
```

ただし、カラーコードを省略したときは、デフォルトのフォアグラウンドカラー(白)で描かれます。

【例題55】 点を描く PSET

画面の中央に点をひとつ描いてください。カラーは青とします。

解答例

```
PSET (320, 200), 1
```

解説 ■ピクセル

グラフィックスの1ドットをピクセルといいます。よーく見ないと青い点が見えないかもしれません。

【例題56】 点を連続して描く PSET

画面の高さ方向中央に点を連続して書くことにより、線を表示してください。ただし線の色は赤とします。

解答例

PC-9801

```
FOR I = 0 TO 639
    PSET (I, 200), 4
NEXT I
```

AXパソコン

```
SCREEN 0
FOR I = 0 TO 639
    PSET(I, 240), 4
NEXT I
```

解説　**■線を描く　LINE**

より速く線を描くコマンドはLINEです。LINEについては、99ページで学習します。

【例題57】　いろいろな色で点を描く　COLOR, PSET

COLOR番号0から7まで(黒～白)の線を交互に画面全体に引いてください。

解答例

```
FOR I = 1 TO 399
    C = I MOD 8
    FOR J = 1 TO 639
        PSET(J, I), C
    NEXT J
NEXT I
```

解説　**■I MOD 8**

MODは、割り算の余りを計算する演算子です。

1 MOD 8は1、2 MOD 8は2、…8 MOD 8は0、9 MOD 8は1…になります。

一定の数値間のサイクリックな数字を取り出すのによく使われます。

■途中で止める

上のプログラムはなかなか終わりません。途中で止める場合は、次のようにしてください。

PC-9801　　[STOP]

AXパソコン　　[Ctrl]＋[Pause]

点を消す　PRESET

すでに描かれている点を消すコマンドは、PRESETです。PRESETの基本書式は次のとおりです。

PRESETの基本書式

```
PRESET (X座標, Y座標)
```

点を描くPSETとこのPRESETを組み合わせると、点を画面上で動かすことができます。

【例題58】 点を消す　PRESET

画面高さ方向中央で、赤い点をスーと動かしてください。

解答例

```
FOR I = 1 TO 639
    PSET (I, 200), 4
    PRESET (I−1, 200)
NEXT I
```

解説　**■点を描いて消す**

PRESETも、正確には点を描くステートメントです。本来の使用方法は次のとおりです。

```
PRESET (X座標, Y座標), カラーコード
```

ただし、この最後のカラーコード(色指定)を省略すると、背景(バックグラウンド)と同じ色を使うために、点が消えたように見えることになります。

■速い！

32bit AXパソコンではとても速く点が動くので、よーく見ていないと分かりません。

線を描く　LINE

線を描くコマンドは、LINEです。LINEは四角を描くのにも使われます。線を描くためのLINEの基本書式は次のとおりです。

LINEの基本書式

```
LINE (開始座標) − (終点座標), カラーコード
```

【例題59】 線を描く　LINE

画面全体にバツ(×)を書いてください。色は緑とします。

解答例

PC-9801

```
LINE (0,0) − (639, 399), 2
LINE (639, 0) − (0, 399), 2
```

AXパソコン

```
LINE (0, 0) − (639, 479), 2
LINE (639, 0) − (0,479), 2
```

【例題60】 線を動かす　LINE

画面の横方向いっぱいの線を、上から下へ動かしてください。

解答例

PC-9801

```
FOR Y = 1 TO 399
    LINE (0, Y) − (639, Y), 2
    LINE (0, Y−1) − (639, Y−1), 0
NEXT Y
```

AXパソコン

```
FOR Y = 1 TO 479
    LINE (0, Y) − (639, Y), 2
    LINE (0, Y−1) − (639, Y−1), 0
NEXT Y
```

解説　■線を消す

線を消すコマンドは用意されていませんので、色を黒にして上書きしてください。

四角を描く　LINE…, B(BF)

四角を描くコマンドは、LINEです。線を描くためのLINEの基本書式にBオプションをつけると四角い枠どりが、またBFオプションをつけると四角の中を塗りつぶすことができます。

LINEの基本書式

```
LINE (左上座標) － (右下座標), カラーコード, B または BF
```

【例題61】　四角を描く　LINE…, B

画面の左上から右下までの一番大きな四角を描いてください。線色は黄色です。

解答例

PC-9801

```
LINE (0, 0) － (639, 399), 6, B
```

AXパソコン

```
LINE (0, 0) － (639, 479), 6, B
```

【例題62】　四角を描いて塗る　LINE…, BF

画面を真っ青にしてください。

解答例

PC-9801

```
LINE (0, 0) － (639, 399), 1, BF
```

AXパソコン

```
LINE (0, 0) － (639, 479), 1, BF
```

解説　■COLOR C1, C2

画面全体を真っ青にするのは、COLORステートメントを使って次のようにすることもできます。ただし、PC-9801シリーズの場合です。

```
COLOR 7, 1
```

AXパソコンでは、COLOR文により画面全体の色を決めることはできません。

円を描く　CIRCLE

円を描くコマンドは、CIRCLE です。CIRCLE では、円弧や扇型を描くこともできます。まず円を描くための基本書式は次のとおりです。

CIRCLE の基本書式

```
CIRCLE （中心座標），半径，色
```

【例題63】　円を描く　CIRCLE

画面中央に、半径100の円を描いてください。色は赤とします。

解答例

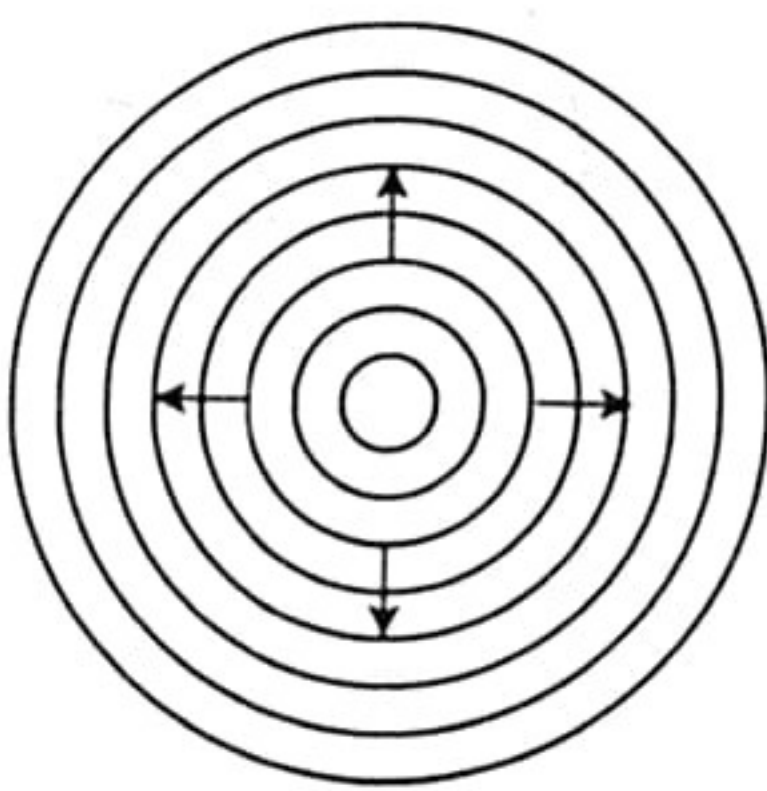
PC-9801

```
CIRCLE （320, 200），100, 4
```

AX パソコン

```
CIRCLE （320, 240），100, 4
```

【例題64】　円を描く　CIRCLE

画面中央から発生して、徐々に広がる円を描いてください。色は緑です。

だんだん大きくなる

解答例

PC-9801

```
FOR I = 1 TO 200
    CIRCLE （320, 200），I, 2
NEXT I
```

AX パソコン

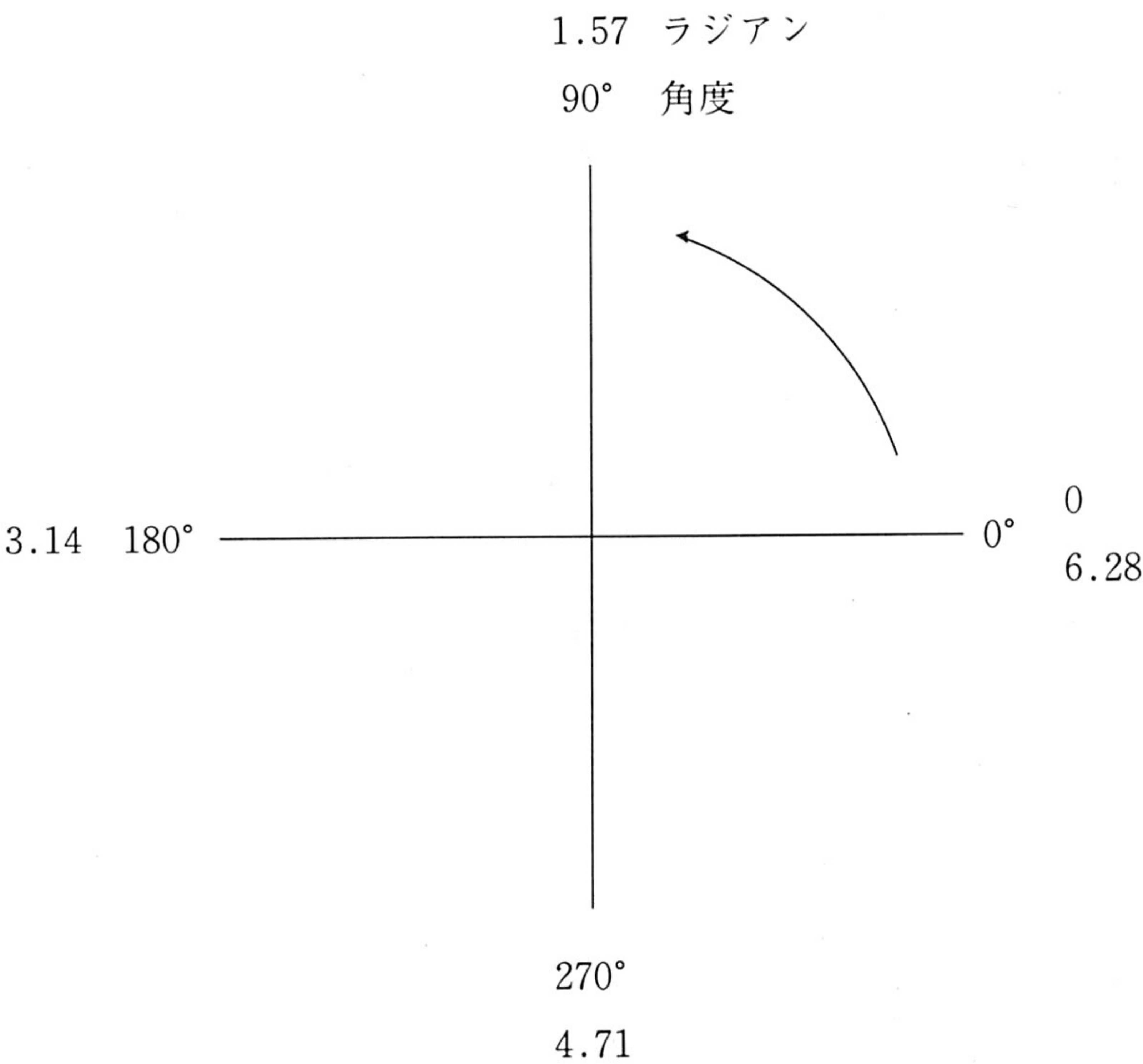
```
FOR I = 1 TO 240
    CIRCLE （320, 240），I, 2
NEXT I
```

円弧を描く　CIRCLE　角度オプション

円弧を描くコマンドはCIRCLEです。円弧は、円の描き始めの角度と終わりの角度をオプションとして指定します。ただし、角度は次のようにラジアンで表現します。角度は、時計の3時の位置から反時計回りに計ります。

1.57　ラジアン
90°　角度

3.14　180°　　0°　0
6.28

270°
4.71

角度からラジアンへは、次の式を使います。

$$\text{ラジアン（単位の角度）} = \frac{\text{「度」単位角度}}{180} \times 3.14$$

円弧を描くためのCIRCLEの基本書式は次のとおりです。

円弧を描く基本書式

```
CIRCLE 中心座標, 半径, 色, 開始角度, 終了角度
```

【例題65】 円弧を描く　CIRCLE　角度オプション

画面の中央に、半径100の1/4円弧を描いてください。ただし色は緑とします。円弧は時計の12時と３時の間に描いてください。

解答例

PC-9801

```
CIRCLE (320, 200), 100, 2, 0, 1.57
```

AX パソコン

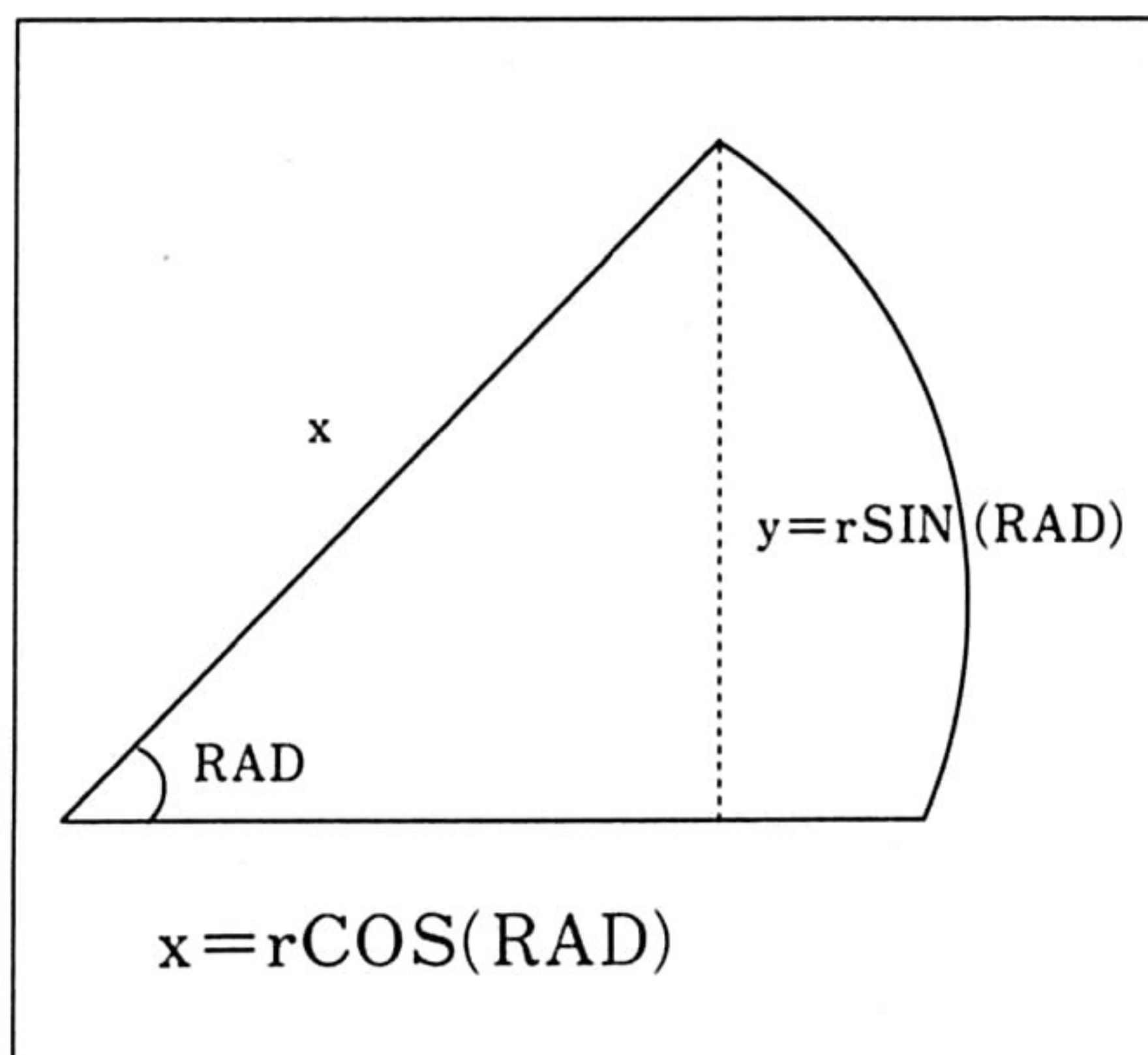
```
CIRCLE (320, 240), 100, 2, 0, 1.57
```

解説　■ラジアンへの変換

90°は、ラジアンでは (90/180) ＊ 3.14 ＝ 1.57 になります。

CIRCLEを使わない円弧の描き型

円を描くステートメントCIRCLEを使わずに円弧を描くにはどうしたらよいでしょうか。もし、三角関数が嫌いでなかったら、次のようにプログラムしてみてください。

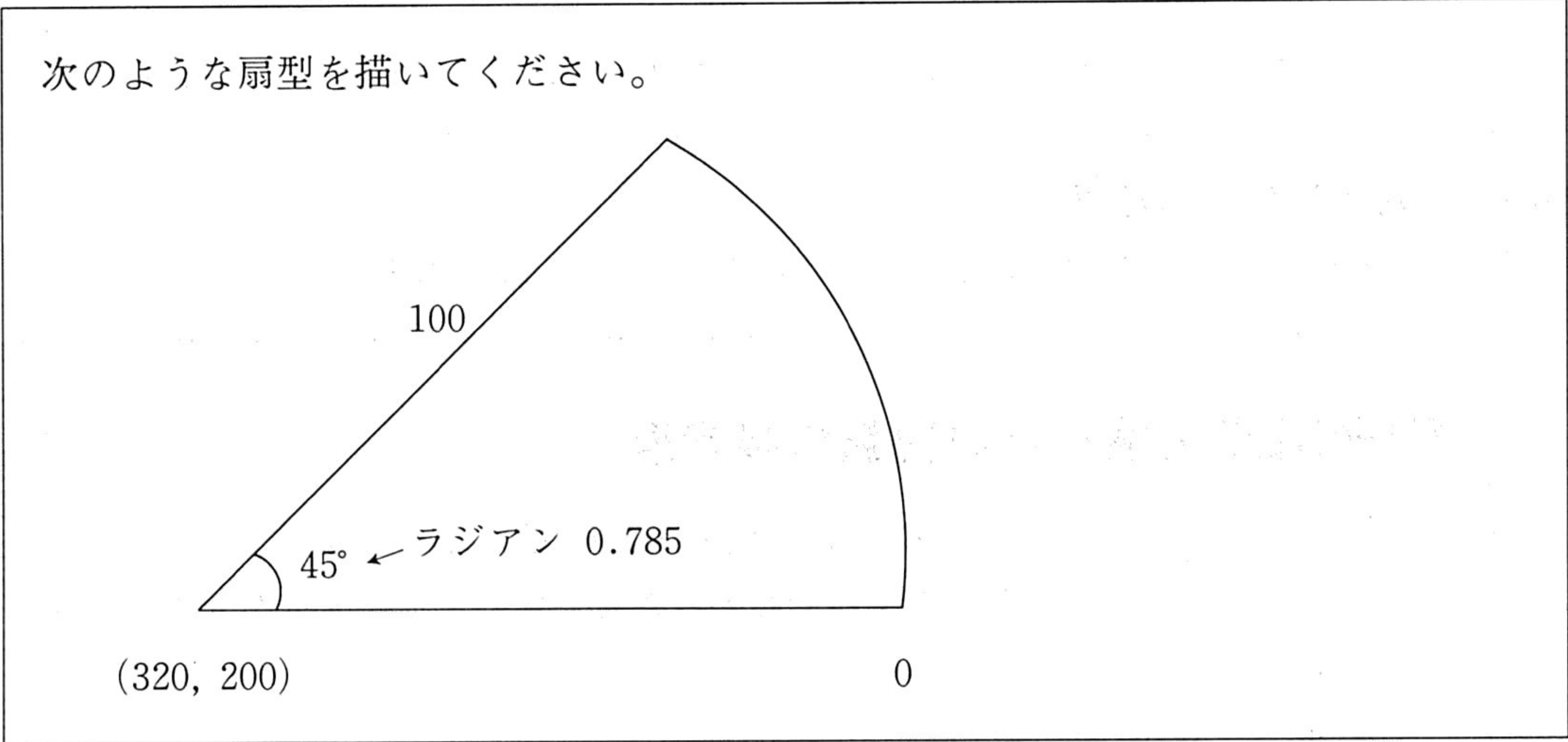
```
FOR C = 0 TO 90
    RAD = C / 180 * 3.14
        PSET(320 + 100 * COS(RAD), 200 - 100 * SIN(RAD)), 2
NEXT C
```


X ＝ 100 ＊ COS(RAD)
Y ＝ 100 ＊ SIN(RAD)

扇型を描く　CIRCLE－角度オプション

扇型を作るコマンドもCIRCLEです。円弧を描く角度指定の数値の前にマイナス記号をつけると扇型になります。しかし、0は－0ではなく－0.001など0に近い0でない数値にマイナスをつけてください。

【例題66】　扇型を描く　CIRCLE　－角度オプション

次のような扇型を描いてください。

解答例

```
CIRCLE (320, 200), 100, 2, -0.001, -1.57
```

解説　**■ラジアンへの変換**

45°は、ラジアンでは (45 / 180) * 3.14 = 0.785 になります。

楕円を描く　CIRCLE 比率オプション

楕円を描く場合は、CIRCLE コマンドの最後にその比率（垂直方向/水平方向）を指定します。

楕円を描く CIRCLE コマンド

```
CIRCLE （中心座標），色，開始角，終了角，比率
```

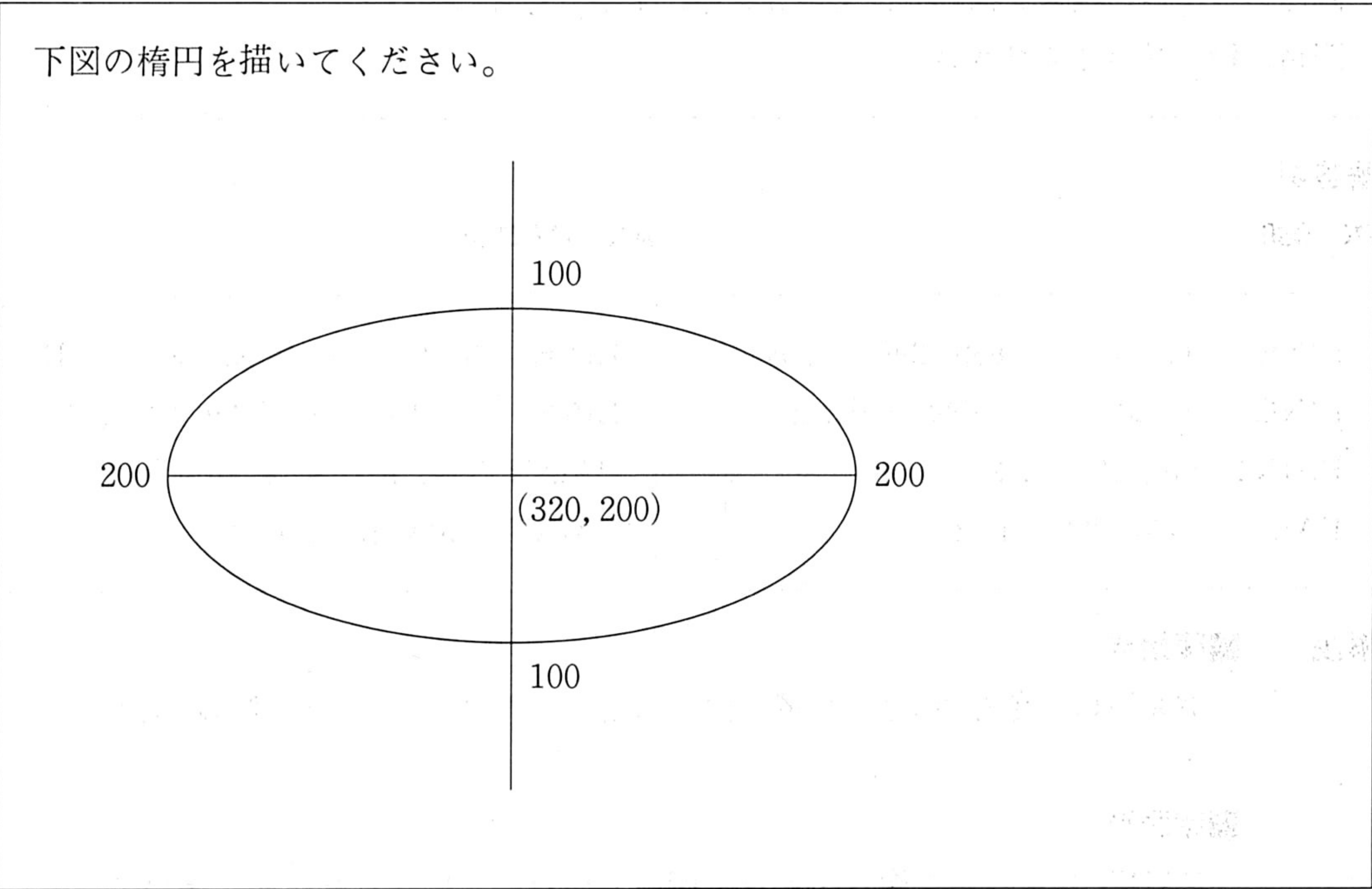
【例題67】 楕円を描く　CIRCLE 比率オプション

下図の楕円を描いてください。

100

200　200

(320, 200)

100

解答例

```
CIRCLE （320, 200），200, 2,,, 0.5
```

解説　**■縦長・横長**

比率１より小さくすると横長、１より大きくすると縦長の楕円になります。

色を塗る PAINT

色を塗るコマンドは、PAINTです。PAINTの基本書式は次のとおりです。

PAINTの基本書式

```
PAINT 開始点, 塗る色, 止まる色
```

【例題68】 色を塗る PAINT

画面いっぱいに四角を描いて、上半分を赤、下半分を青に塗ってください。
四角は緑で描いてください。

解答例

PC-9801

```
LINE (0, 0) - (639, 399), 2, B
LINE (0, 200) - (639, 200), 2
PAINT (10, 10), 4, 2
PAINT (400, 300), 1, 2
```

AXパソコン

```
LINE (0, 0) - (639, 479), 2, B
LINE (0, 240) - (639, 240), 2
PAINT (10, 10), 4, 2
PAINT (400, 300), 1, 2
```

解説

■開始点

開始点は、色を塗る図形の内側の点を指定してください。外枠の線上ではだめです。

■閉図形

PAINTにより色を塗る図形は、閉じていなくてはなりません。開いていると、そこから色がもれます。ただし、図形の外側を塗る場合には、ディスプレイの端で色塗りは止まります。

色を変える　PALETTE

PSET や LINE などのカラー番号を指定して描いた図形の色、PAINT によって塗った色を、一瞬にして変えることができます。95ページのカラー番号と色の対応を変えると、すでに描かれている図形の色が変わります。

いままで1は青だったのを、1は赤とすれば、青い図形は赤くなります。

PALETTE

色番号と色の関係を再設定するステートメントは PALETTE です。

PALETTE の基本書式

```
PALETTE　カラー番号, 対応カラー
```

【例題69】　色を変える　PALETTE

左上を(10, 10)、右下を(200, 200)とする、内部も青の四角を描き、しばらくたつとその色が赤に変わるプログラムを作成してください。

解答例

```
CLS
LINE(10, 10)　－　(200, 200), 1, BF
FOR　I　＝　1　TO　10000
NEXT　I
PALETTE　1, 4
```

色を調べる　POINT

ある点の色が何色かは、POINT 関数で調べることができます。

POINT

POINT の基本書式

```
POINT(座標)
```

指定した座標点のカラーコードが返されます。

【例題70】　色を調べる　POINT

画面の中央に縦の青い線を1本引いて、左から赤い点を動かし線を描き、青い線を横切ったら黄色い線に変えてください。

解答例

```
LINE(320, 0) - (320, 399), 1
C = 4
FOR X = 1 TO 639
    PSET(X, 200), C
    IF POINT(X + 1,200) = 1 THEN
        C = 6
    END IF
NEXT X
```

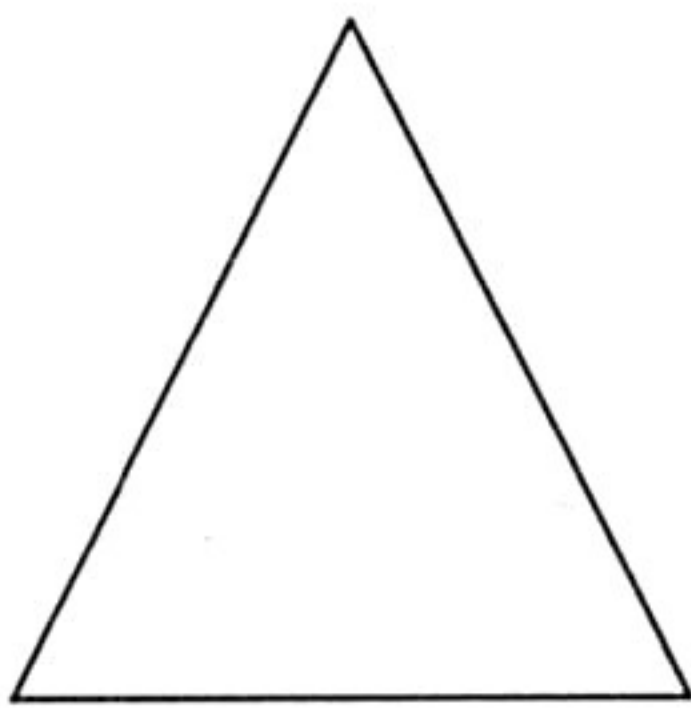
解説　■ひとつ先を見る　X + 1

POINT 関数で調べる色は、動く方向のひとつ先です。動いている点は自分の色ですから、青を検知できません。

連続描画　DRAW

Quick BASICには、点を移動させながら連続的に描画していくDRAWステートメントがあります。図形を回転させたり、拡大縮小できる多機能なコマンドですが、基本的な連続描画の使い方を学習します。

DRAW

DRAWステートメントの基本的な使い方は、次のとおりです。

DRAWの基本書式

```
DRAW "方向　移動量"
   方向 U 上    E 右斜め上
        D 下    F 右斜め下
        L 左    G 左斜め下
        R 右    H 左斜め上
```

ただし、描き始めの点は、DRAW "M 座標" で決定されます。

【例題71】 連続描画　DRAW

画面中央に、底辺が200の直角二等辺三角形を描いてください。

200

解答例

PC-9801

```
DRAW "M 420, 200"
DRAW "H 100"
DRAW "G 100"
DRAW "R 200"
```

AXパソコン

```
DRAW "M 420, 240"
DRAW "H 100"
DRAW "G 100"
DRAW "R 200"
```

グラフィック範囲の限定　VIEW

ビューポート

画面の一部分についてのみ、グラフィックを描く範囲として設定することができます。画面の一部でグラフィックが描かれる窓のことを、ビューポートといいます。

ビューポートの設定は、VIEW ステートメントで行ないます。

VIEW の基本書式

```
VIEW(左上の座標) － (右下の座標)[, 内部の色][, 境界の色]
```

内部の色と境界の色はオプションです。

相対座標

ビューポートの中では、ビューポート左上を(0，0)とする相対的な座標を使ってください。

【例題72】　グラフィック範囲の限定　VIEW

画面の (100, 100) と (450, 300) を対象点とする四角領域を、ビューポートとして設定してください。その中で座標 (100, 100) を中心とする円を半径50から拡げていき、ビューポートの外には描画されないことを確認してください。

解答例

```
VIEW(100, 100) － (450, 300),,4
FOR I = 50 TO 260
    CIRCLE(100, 100), I, 2
NEXT I
```

第5章　データ

第5章では、多量のデータを扱う手法として、READ～DATA文と、同種のデータを効率よく扱う配列について学習します。異なる型のデータを総合的に扱うレコードの考え方については、第3部で学習します。

第5章の概要

■プログラムからのデータの入力　READ ～ DATA

プログラムの中にDATAとして書いたデータをREADで順次読み取ることができます。

■配列　変数名(n)

同種のデータは配列として添字で区分できます。

■配列の宣言　DIM

要素数が10を超す配列は、DIMで宣言する必要があります。

■2次元配列　DIM…(…, …)

表の形式のデータを配列として扱うには、2次元配列を使います。2次元配列も、DIMによる使用宣言が必要です。

プログラムからのデータ入力　READ ～ DATA

データが固定的で多くの量があるときは、そのデータそのものをプログラムの中に書いて処理することができます。

データ行

プログラムの中で、その行がDATAで始まる行は、READ文に対するデータ行とみなされます。READは、1回呼ばれる度にDATA行からデータを読み込んできます。

READ～DATAの基本書式

```
READ　変数名, [変数名, …]

            ⋮

DATA　データ[, データ…]
```

【例題73】 プログラムからのデータ入力　READ ～ DATA

A君の期末テストの成績は、次のとおりでした。5教科の平均点を計算してください。
82, 76, 44, 92, 100

解答例 (1)

```
READ A
READ B
READ C
READ D
READ E
SUM = A + B + C + D + E
AVE = SUM / 5
PRINT AVE
DATA 82, 76, 44, 92, 100
```

解答例 (2)

```
SUM = 0
FOR I = 1 TO 5
    READ A
    SUM = SUM + A
NEXT I
AVE = SUM / 5
PRINT AVE
DATA 82, 76, 44, 92, 100
```

解説　■1対1の対応

上の解答例(1)では、変数Aに82、Bに76、Cに44…が入ります。

【例題74】 プログラムからのデータ入力　READ ～ DATA

次の10個のデータの内最大のものを表示してください。
100, 15, 200, 201, 416, 781, 20, 80, 91, 415

解答例

```
READ  MAX
FOR  I  =  1  TO  9
       READ  B
       IF  B  >  MAX  THEN
             MAX  =  B
       END  IF
NEXT  I
PRINT  MAX
DATA  100, 15, 200, 201, 416, 781, 20, 80, 91, 415
```

解説　**■MAXの初期値**

MAXの最初の値は、READ MAXによりひとつ目のデータ100に設定されます。それ以降、READのたびにB ＝ 15,200… などと大小の比較が行なわれ、常にその時点で最大が保持されています。

【例題75】 文字型データの読み込み　READ ～ DATA

次の５人の野球選手の名前を、１行ごとに表示してください。
長島、達川、落合、岡田、原

解答例

```
FOR I = 1 TO 5
    READ A$
    PRINT A$
NEXT I
DATA 長島, 達川, 落合, 岡田, 原
```

解説　■**文字列**

READ～DATA 構文では、文字列を読み込むこともできます。DATA には、""は不要です。

【例題76】 複数データの読み込み　READ ～ DATA

次の5組のデータについて、それぞれ大きい数値を表示するプログラムを作成してください。

データ　2, 10
　　　100, 80
　　　42, 36
　　　18, 19
　　　42, 44

解答例

```
FOR I = 1 TO 5
    READ A, B
    IF A < B THEN
        A = B
    END IF
    PRINT A
NEXT I
DATA 2, 10, 100, 80, 42, 36, 18, 19, 42, 44
```

解説　■**2つずつ**

READ 文は、その引数の個数だけ一度にデータを読みます。上の例では、同時に2つのデータを読みます。

【例題77】 図形データの読み込み　READ ～ DATA

次の3つの円を描いてください。

中心	半径	枠色	内部色
320, 200	100	赤(4)	緑(2)
100, 100	50	白(7)	黄(6)
500, 200	40	青(1)	紫(3)

解答例

```
FOR I = 1 TO 3
    READ X, Y, R, C1, C2
    CIRCLE(X, Y), R, C1
    PAINT(X, Y), C2, C1
NEXT I
DATA 320, 200, 100, 4, 2
DATA 100, 100, 50, 7, 6
DATA 500, 200, 40, 1, 3
```

解説　**■パラメータの多いステートメント**

READ～DATAによる構文は、グラフィックのようにパラメータの多いステートメントを繰り返し使うときに、大きな力を発揮します。

データの読み直し RESTORE

RESTORE

READ～DATA によって一度読んだデータをもう一度読み直す場合は、RESTORE ステートメントを用います。RESTORE によって、プログラムの最初の DATA 行に読み取りポイントがセットされます。

【例題78】 データの読み直し RESTORE

次の各座標をそれぞれ左下、右下の座標とする四角形を、青から白まで7回分描いてください。ただし、それぞれの描画の間に少しの時間をおいて、その変化が分かるようにしてください。

左上	右下
(0, 0)	(100, 100)
(150, 150)	(300, 300)
(301, 301)	(639, 399)

解答例

```
FOR J=1 TO 7
    FOR K=1 TO 3
        READ X1, Y1, X2, Y2
        LINE (X1, Y1)-(X2, Y2), J, BF
    NEXT K
    FOR L=1 TO 1000
    NEXT L
    RESTORE
NEXT J
DATA 0, 0, 100, 100
DATA 150, 150, 300, 300
DATA 301, 301, 639, 399
```

配列　変数名(n)

配列

同種のデータを多量に扱うときは、そのデータを表す変数名の次に( )を付け、その番号で区分することがあります。

人数(1)	午前中	100人
人数(2)	午後	200人
人数(3)	合計	300人

要素数・要素番号

変数名の次に( )を付けて、同種のデータを区分するものを配列といいます。

( )内の各数値を配列の要素番号、その最大値を配列の要素数といいます。

【例題79】　配列　変数名(n)

出席番号1～5までの生徒の名前は次のとおりです。出席番号を入力すると生徒の名前が出てくるプログラムを作成してください。

1　青木
2　井上
3　牛島
4　江原
5　大西

解答例 (1)

```
NAMAE$(1) = "青木"
NAMAE$(2) = "井上"
NAMAE$(3) = "牛島"
NAMAE$(4) = "江原"
NAMAE$(5) = "大西"
INPUT "番号をどうぞ", I
PRINT NAMAE$(I)
```

解答例 (2)

```
FOR I = 1 TO 5
    READ NAMAE$(I)
NEXT I
INPUT "番号をどうぞ", I
PRINT NAMAE$(I)
DATA 青木, 井上, 牛島, 江原, 大西
```

配列の宣言　DIM

要素数が10個までの配列は、特別な宣言なしで使うことができますが、10を超す配列やこの次に学習する2次元以上の配列では、特別に宣言が必要です。配列の宣言は、DIM 宣言子を使います。

DIM

配列の宣言

```
DIM　変数名(要素数)
```

5つの配列を宣言するためにDIM　X(5)としたときには、実際にはX(0)という要素を含めてX(0), X(1), …X(5)の6個になります。しかし、思考の上で0番目というのはネックになりますので、少しもったいないのですが、0番目は使わないで1から数えて5個ということにしてください。

【例題80】　配列の宣言　DIM

次の20人分の成績データを配列にしまい、そののちに平均値を求めてください。

データ　82, 84, 86, 92, 48, 72, 61, 61, 43, 99, 77, 75, 41, 32, 97, 66, 62, 55, 41, 100

解答例

```
DIM SEISEKI(20)
SUM = 0
FOR I = 1 TO 20
    READ SEISEKI(I)
    SUM = SUM + SEISEKI(I)
NEXT I
AVE = SUM / 20
PRINT AVE
DATA 82, 84, 86, 92, 48, 72, 61, 61, 43, 99, 77, 75, 41, 32, 97, 66, 62, 55, 41, 100
```

2次元配列　DIM…(…, …)

2次元配列

配列において、添え字を2個使って表形式のデータを処理することもできます。2つの添え字を使った配列を2次元配列といいます。これもやはりDIMによる配列宣言が必要です。

配列宣言(2次元)

```
DIM　変数名(要素数1, 要素数2)
```

【例題81】　2次元配列　DIM…(…, …)

次の表の横の合計を求めてください。

	国語	算数	理科	社会	英語	合計
青木	82	83	42	91	100	
井上	88	89	63	100	85	
牛島	92	88	72	91	66	
江原	61	42	38	61	72	

解答例

```
DIM S(4, 6)
FOR I = 1 TO 4
    FOR J = 1 TO 5
        READ S(I, J)
        S(I, 6) = S(I, 6) + S(I, J)
    NEXT J
    PRINT S(I, 6)
NEXT I
DATA 82, 83, 42, 91, 100
DATA 88, 89, 63, 100, 85
DATA 92, 88, 72, 91, 66
DATA 61, 42, 38, 61, 72
```

解説　**■Ｉは縦、Ｊは横**

左ページの解答例で、カウント変数ＩとＪは、それぞれ表の縦と横に対応します。

■合計欄

横の合計を入れるために、横方向はデータ数よりひとつ多く配列宣言しています。

【例題82】 配列データの検索

次のデータは名前と平均得点です。名前を入力すると平均得点を答えるプログラムを作成してください。

青木	82.6
井上	79.1
牛島	92.8
江原	72.1
大西	95.6

解答例

```
FOR I = 1 TO 5
    READ A$(I), B(I)
NEXT I
INPUT "名前をどうぞ", NAMAE$
FOR I = 1 TO 5
    IF A$(I) = NAMAE$ THEN
        EXIT FOR
    END IF
NEXT
PRINT B(I)
DATA 青木, 82.6, 井上, 79.1, 牛島, 92.8, 江原, 72.1, 大西, 95.6
```

解説　**■１次元配列を２つ**

文字型データと数値型データですから、ひとつの２次元配列に入れることができません。１次元配列を２つ使って処理します。

第6章　関　数

Quick BASIC には、プログラムの基本的な要素として関数が多く用意されています。

本章では、Quick BASIC の多くの関数の中から、よく使う関数を取り上げて解説します。

関数とは、与えられた数値やイベント(キーが押された、カーソルが動かされた…)などの処理をして、その処理結果を数値や文字列などの値として返すものです。

第６章の概要

■算術関数

切り捨て	INT
乱数	RND

■日付と時刻

日付	DATE$
時刻	TIME$

■文字列操作関数

部分文字列	LEFT$
	MID$
	RIGHT$
繰り返し文字列	STRING$
文字列の長さ	LEN
空白	SPACE$

■キーセンス　INKEY$

■文字列と数値の変換　VAL, STR$

■文字コードとその操作　ASC, CHR$

■アスキーコード表

算術関数 INT, RND

関数といえば、数学の世界です。まず最初にQuick BASICに備わっている算術関数のいくつかを使ってみましょう。

切り捨て

関数　INT
機能　小数点以下の切り捨て
書式　INT(式)
使用例　A = INT(10 /3)
　　　　　PRINT A
　　　　　（3が表示される）

【例題83】　切り捨て　INT

給料(最大999,999円)をキーボードから入力すると、その金種(10000円札何枚、5000円札何枚…)を表示するプログラムを作成してください。

解答例

```
INPUT "金額", ZAN
FOR I = 1 TO 9
    READ N
    X = INT(ZAN / N)
    PRINT N; "円は"; X;"枚です"
    ZAN = ZAN - X * N
NEXT
DATA 10000, 5000, 1000, 500, 100, 50, 10, 5, 1
```

解説　**■金種計算**

金種計算は、給与計算プログラムなどにはつきものです。

乱数　**関数**　RND
　　　機能　０から１までの乱数
　　　書式　RND
　　　使用例　A ＝ 10 ＊ RND

【例題84】　乱数　RND

乱数を10回発生させて表示してください。

解答例

```
FOR I = 1 TO 10
    PRINT RND
NEXT I
```

【例題85】　乱数の整数化

２から７までの整数の乱数を10回発生させてください。

解答例

```
FOR I = 1 TO 10
    A = INT(6 * RND) + 2
    PRINT A
NEXT
```

解説　**■N1からN2の乱数（整数）**

RND関数は０～１の間の乱数を発生させますので、整数N1からN2までの間の乱数は次の式で与えられます。

```
INT((N2 - N1 + 1) * RND) + N1
```

日付と時刻　DATE$, TIME$

計算機に内蔵されているタイマーから、日付と時刻を取り出すことができます。

日付　**関数**　DATE$
機能　日付を文字列として取り出す
書式　DATE$
使用例　PRINT　DATE$　→　1989-05-19　と表示

時刻　**関数**　TIME$
機能　現在の時刻を文字列として取り出す
書式　TIME$
使用例　PRINT　TIME$　→　09:27:12　と表示

【例題86】　日付と時刻　DATE$, TIME$

今日の日付と時刻を表示してください。

解答例

```
CLS
PRINT  DATE$
PRINT  TIME$
```

日付と時刻の設定

DATE$とTIME$は、日付と時刻を設定するためにも用いられます。

日付の設定

DATE$　=　"1989-05-25" ⏎

時刻の設定

TIME$　=　"20:58:12" ⏎

文字列操作関数 LEFT$, RIGHT$, …

Quick BASIC には、文字列を操作する関数が多数あります。代表的ないくつかを学習しましょう。

部分文字列の取り出し

関数　LEFT$
機能　ある文字列の左から何文字かを取り出す
書式　LEFT$(元の文字列，文字数)
使用例

```
A$ = LEFT$("ABCDEF", 3)
PRINT A$
```

→ ABC と表示

関数　RIGHT$
機能　ある文字列の右から何文字かを取り出す
書式　RIGHT$(元の文字列，文字数)
使用例

```
A$ = RIGHT$("ABCDEF", 3)
PRINT A$
```

→ DEF と表示

関数　MID$
機能　ある文字列の任意の箇所、長さの文字列を取り出す
書式　MID$(元の文字列，開始位置，長さ)
使用例

```
A$ = MID$("ABCDEF", 2, 3)
PRINT A$
```

→ BCD と表示

【例題87】　部分文字列　LEFT$, MID$, RIGHT$

> 今日の日付を西暦△年△月△日と表示してください。

解答例

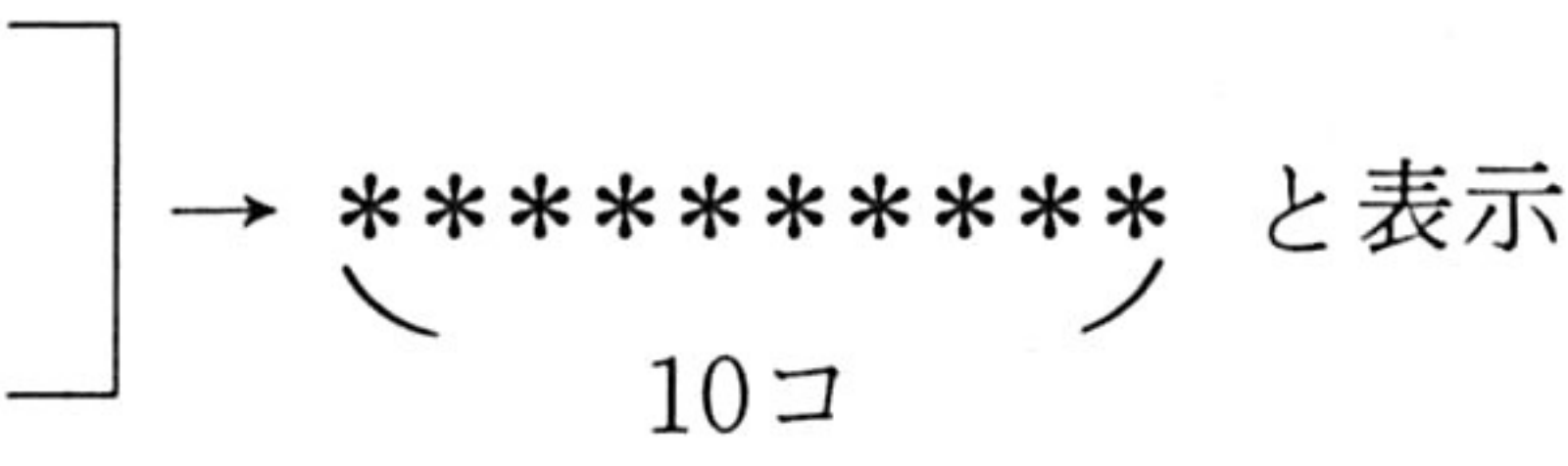
```
CLS
A$ = DATE$
YY$ = LEFT$(A$, 4)
MM$ = MID$(A$, 6, 2)
DD$ = RIGHT$(A$, 2)
PRINT "西暦"; YY$;"年"; MM$;"月"; DD$;"日"
```

文字の繰り返し

関数　STRING$

機能　文字を繰り返して文字列にする

書式　STRING$(回数, 繰り返す文字)

使用例　A$ = STRING$(10, "*")
　　　　PRINT A$
→ ********** と表示（10コ）

【例題88】 文字の繰り返し STRING$

画面の最上行に = を80個繰り返して書いてください。

解答例

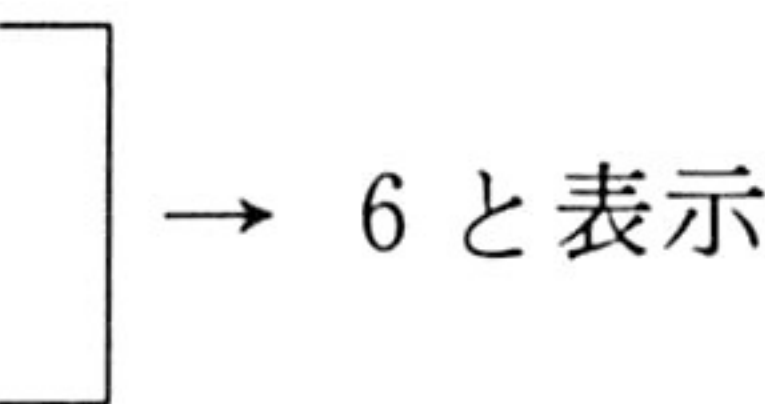
```
CLS
LOCATE 1, 1
PRINT STRING$(80, "=")
```

文字列の長さ

関数　LEN

機能　文字列の長さを調べる

書式　LEN(文字列)

使用例　N = LEN("ABCDEF")
　　　　PRINT N
→ 6 と表示

【例題89】 文字列の長さ　LEN

入力された文字列が10桁ない場合は、その後を＊でうめるプログラムを作成してください。

解答例

```
INPUT "文字列", A$
B$ = ""
IF LEN(A$) < 10 THEN
    B$ = STRING$(10 - LEN(A$), "*")
END IF
PRINT A$ + B$
```

空白

関数　SPACE$

機能　指定された個数のスペースを発行する

書式　SPACE$(個数)

使用例

```
A$ = "ABC"
B$ = "DEF"                        → ABC     DEF と表示
PRINT A$ + SPACE$(5) + B$
```

【例題90】 空白　SPACE$

入力された名前を、赤の反転で次のように表示してください。

```
[              山田 太郎]  右詰め
      反転領域20文字分
```

解答例　(　)内はAXパソコンでの解答例

```
CLS
INPUT "名前", NAMAE$
COLOR 12                    (COLOR 0, 4)
PRINT SPACE$(20 - LEN(NAMAE$)) + NAMAE$
COLOR 7                     (COLOR 7, 0)
```

キーセンス INKEY$

データを入力するINPUTは、リターンが押されるまで入力待ちの状態でスタンバイします。その瞬間INPUTのように停止せずに、キーが押されているか、もし押されていればその値を返す関数INKEYがあります。キーの押下をセンス(調べる)する関数として重要です。

キーセンス

関数　INKEY$

機能　キーが押下されているかのチェック。キーが押下されていないときは、ヌル文字("")を返す。

書式　INKEY$

使用例

```
WHILE  INKEY$  =  ""   ┐
    ⋮                  ├→ キーが押されるまで⋮を繰り返す
WEND                   ┘
```

注意　単独で押す SHIFT や CTRL キーはセンスされません。

【例題91】　キーセンス　INKEY$

「何かキーを押してください」と表示して、プログラムの進行を一時的に止めるプログラムを作成してください。

解答例(1)

```
PRINT  "何かキーを押してください"
DO
LOOP  UNTIL  INKEY$  <>  ""
```

解答例(2)

```
PRINT  "何かキーを押してください"
WHILE  INKEY$  =  ""
WEND
```

解答例(3)

```
PRINT  "何かキーを押してください"
DO
LOOP  WHILE  INKEY$  =  ""
```

文字列と数値の変換 VAL, STR$

数字からなる文字列データを数値に、逆に数値をその数字からなる文字列に変換することができます。

文字列→数値

関数	VAL	
機能	文字列を数値に変換する	
書式	VAL(文字列)	数字と小数点からなる文字列
使用例	N = VAL("123")	Nは123という数値

数値→文字列

関数	STR$	
機能	数値を文字列に変換する	
書式	STR$(数値)	
使用例	S = STR$(123)	Sは"123"という文字列

【例題92】　文字列←→数値　VAL, STR$

キーボードから入力された文字列を数値になおして、2で割って再び文字列として表示してください。

解答例

```
INPUT "文字列", A$
N = VAL(A$) / 2
PRINT STR$(N)
```

文字コードとその操作 ASC, CHR$

カラーに対応するカラー番号(カラーコード)があるように、文字にも一文字一文字番号が付いています。これを、文字コードといいます。

アスキーコード

半角の英数、カタカナ、グラフィック文字などと一部の特殊記号には、0から255までの番号が付いています。これが、通常アスキーコードといわれる文字番号です。

アスキーコードと文字の対応表を次々頁に示します。アスキーコード0から31までは、文字にならない制御コードですから省略してあります。

文字のアスキーコード

文字のアスキーコードは、関数ASCで調べることができます。

関数 ASC

機能 文字のアスキーコードを返す

書式 ASC(文字)

使用例
```
N = ASC("A")  ─┐
               ├→ 65と表示
PRINT N       ─┘
```

【例題93】 アスキーコード ASC

> 入力された文字が大文字ならブザーを鳴らすプログラムを作成してください。

解答例

```
INPUT "一文字入力 リターン", A$
IF ASC(A$) >= 65 AND AS(A$) <= 90 THEN
    BEEP
END IF
```

アスキーコードの文字化

関数ASCとは逆に、与えられたアスキーコードから文字を作り出す関数CHR$があります。

関数　CHR$

機能　アスキーコードの文字化

書式　CHR$(アスキーコード)

使用例　PRINT　CHR$(233) → ♥(ハート) が表示される

【例題94】　アスキーコードの文字化　CHR$

スペード、ハート、ダイヤ、クラブの記号を表示させてください。

解答例

```
FOR I = 232 TO 235
    PRINT CHR$(I) + " "
NEXT I
```

【例題95】　アスキーコードの文字化　CHR$

入力された文字が大文字なら、小文字にして表示してください。

解答例

```
INPUT "一文字 リターン", A$
IF ASC(A$) >= 65 AND ASC(A$) <= 90 THEN
    A$ = CHR$(ASC(A$) + 32)
END IF
PRINT A$
```

アスキーコード表

番号	記号	番号	記号	番号	記号	番号	記号	番号	記号	番号	記号	番号	記号
32	(SP)	64	@	96		128	▁	160	(SP)	192	タ	224	═
33	!	65	A	97	a	129	▂	161	。	193	チ	225	╞
34	”	66	B	98	b	130	▃	162	「	194	ツ	226	╪
35	#	67	C	99	c	131	▄	163	」	195	テ	227	╡
36	$	68	D	100	d	132	▅	164	、	196	ト	228	◢
37	%	69	E	101	e	133	▆	165	・	197	ナ	229	◣
38	&	70	F	102	f	134	▇	166	ヲ	198	ニ	230	◥
39	’	71	G	103	g	135	█	167	ァ	199	ヌ	231	◤
40	(	72	H	104	h	136	▏	168	ィ	200	ネ	232	♠
41	)	73	I	105	i	137	▎	169	ゥ	201	ノ	233	♥
42	*	74	J	106	j	138	▍	170	ェ	202	ハ	234	♦
43	+	75	K	107	k	139	▌	171	ォ	203	ヒ	235	♣
44	,	76	L	108	l	140	▋	172	ャ	204	フ	236	●
45	−	77	M	109	m	141	▊	173	ュ	205	ヘ	237	○
46	.	78	N	110	n	142	▉	174	ョ	206	ホ	238	／
47	/	79	O	111	o	143	┼	175	ッ	207	マ	239	＼
48	0	80	P	112	p	144	┴	176	ー	208	ミ	240	×
49	1	81	Q	113	q	145	┬	177	ア	209	ム	241	円
50	2	82	R	114	r	146	┤	178	イ	210	メ	242	年
51	3	83	S	115	s	147	├	179	ウ	211	モ	243	月
52	4	84	T	116	t	148	▔	180	エ	212	ヤ	244	日
53	5	85	U	117	u	149	─	181	オ	213	ユ	245	時
54	6	86	V	118	v	150	│	182	カ	214	ヨ	246	分
55	7	87	W	119	w	151	▕	183	キ	215	ラ	247	秒
56	8	88	X	120	x	152	┌	184	ク	216	リ	248	
57	9	89	Y	121	y	153	┐	185	ケ	217	ル	249	
58	:	90	Z	122	z	154	└	186	コ	218	レ	250	
59	;	91	[	123	{	155	┘	187	サ	219	ロ	251	
60	<	92	¥	124	|	156	╭	188	シ	220	ワ	252	
61	=	93	]	125	}	157	╮	189	ス	221	ン	253	
62	>	94	^	126	~	158	╰	190	セ	222	゛	254	
63	?	95	_	127		159	╯	191	ソ	223	゜	255	

第 3 部

プログラミングⅡ（中級編）

第1章　プロシージャ

第1章では、大きなプログラムを作成するときに役立つプロシージャについて学習します。一度作成したプロシージャは、別のプログラムを作成するときにも使える組立部品(ビィルディングブロック)になります。

第1章の概要

■プロシージャ

プロシージャには、

(1) 関数

(2) サブプログラム

の2つがあります。

■プロシージャの作成と呼び出し

プロシージャの作成から呼び出しまでは、一連のメニュー操作で実行できます。

■関数の定義　FUNCTION ～ END FUNCTION

関数の定義は、FUNCTIONで始まりEND FUNCTIONで終わります。

■サブプログラムの定義　SUB ～ END SUB

サブプログラムの定義は、SUBで始まりEND SUBで終わります。

■サブファイルの常駐

他のプログラムからプロシージャを使うときは、プロシージャを含むサブファイルを常駐させておく必要があります。

■プロシージャの呼び出し

プロシージャは、パラメータを与えて呼び出します。サブプログラムは、CALL文を使います。

■メインモジュールの保存

メインモジュールには、独立したファイル名を与えて保存してください。

■プロシージャの編集

プロシージャは、メニュー[V/表示]の機能を使って直接編集できます。

■プロシージャと変数

プロシージャの仮引数は、ローカル変数です。また、実引数は「参照による呼び出し」形式で、その値が使われます。

プロシージャ

ビルディングブロック

Quick BASICでは、プログラムのビルディングブロック(組立部品)として、プロシージャが使えます。プロシージャは、よく使う機能をまとめて名前を付けて再利用するものです。プロシージャには、次の2つのものがあります。

プロシージャ

関数 サブプログラム

関数　関数は、呼び出されるとその内容を実行し、その結果を値として返します。

(例)　2つの数値のうち大きい方を返す関数

```
FUNCTION MAX(A, B)
IF A > B THEN
    MAX = A
ELSE
    MAX = B
END IF
END FUNCTION
```

呼び出して使う側

```
  PRINT MAX(12, 25)
```

サブプログラム

サブプログラムは、呼び出されるとその内容を実行します。ただし、値は返しません。

(例)　2つの数値の大きい方を表示するサブプログラム

```
SUB MAX(A, B)
IF A > B THEN
    PRINT A
ELSE
    PRINT B
END IF
END SUB
```

呼び出して使う側

```
  MAX(12, 25)
```

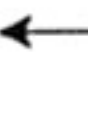
プロシージャの作成と呼び出し

プロシージャを作成して呼び出すまでの全体手順を説明します。

(1)サブファイルの作成

プロシージャを集めたサブファイルを作成します。

(2)メインファイルの作成

＜サブファイルの常駐＞

サブファイルを、呼び出しにそなえて常駐させます。

＜メインプログラムの記述＞

メインプログラムの中で、プロシージャを呼び出します。

以下に2つの数の内大きい方を返す関数MAXと、小さい方を返す関数MINの作成例を示します。次のセクションから例題としてステップごとに実行しますので、ここでは読み通すだけにしてください。

(1)サブファイルの作成

```
新規ファイル　[GRPH],[F],[C](AXパソコン　[Alt],[F],[C])
ファイル名の指定　FUNCS [↵]
FUNCTION  MAX(A, B)
IF  A  >  B  THEN
      MAX  =  A
ELSE
      MAX  =  B
END  IF
END  FUNCTION
FUNCTION  MIN(A, B)   ←ここで[↵]をすると、画面は新しい
IF  A  <  B  THEN        関数MIN(A, B)の定義画面になります。
      MIN  =  A
ELSE
      MIN  =  B
END  IF
END  FUNCTION
ファイルの保存　[GRPH],[F],[S](AXパソコン　[Alt],[F],[S])
```

(2)メインファイルの作成

(イ)新規ファイルのオープン GRPH, F, N (AX パソコン Alt, F, N)

(ロ)サブファイルの常駐

PC-9801

```
GRPH, F, L
FUNCS ↵
```

AX パソコン

```
Alt, F, L
FUNCS ↵
```

実行例

```
DECLARE FUNCTION MAX!(A!, B!)
DECLARE FUNCTION MIN!(A!, B!)
```

(ハ)メインプログラムの記述

```
PRINT MAX(12, 25)
PRINT MIN(12, 25)
```

(3)メインプログラムの実行 SHIFT+f·5

```
25 ← PRINT MAX の実行結果
12 ← PRINT MIN の実行結果
```

(4)メインプログラムの保存(名前を変えて保存)

```
GRPH        (AX パソコン Alt)
F
A
TEST ↵
```

関数の定義　FUNCTION ～ END FUNCTION

関数の定義は、FUNCTION～END FUNCTIONで行ないます。Quick BASICのエディットモードでは、FUNCTION 関数名⏎と入力するとEND FUNCTIONが自動的に表示されます。

キーワードFUNCTIONに続けて、そのFUNCTIONが使われる形式を示す関数名を書きます。行を改めてその関数の内容を記述することになります。

関数の一般形

```
FUNCTION 関数名(引数, …)
定義
END FUNCTION
```

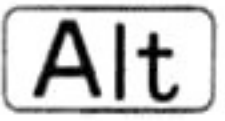
【例題96】 関数の定義　FUNCTION ～ END FUNCTION

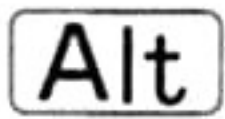
新しくサブファイルFUNCSを作成し、2つのうち大きい数値を返す関数MAX(A, B)を作成して、ファイル名FUNCSでセーブしてください。

解答例

```
GRPH (AXパソコン Alt)
F
C
ファイル名 FUNCS ⏎
FUNCTION MAX(A, B)
IF A > B THEN
    MAX = A
ELSE
    MAX = B
END IF
END FUNCTION
GRPH (AXパソコン Alt)
F
S
```

サブプログラムの定義　SUB ～ END SUB

サブプログラムの定義は、SUB～END　SUBで行ないます。Quick　BASICのエディットモードでSUB　プログラム名⏎と入力すると、対応するEND　SUBが自動的に表示されます。

サブプログラムの一般形

```
SUB　サブプログラム名(引数, …)
定義
END　SUB
```

【例題97】　サブプログラムの作成　SUB ～ END SUB

与えられた文字列を表示するときに、「私の名前は」をつけて表示するサブプログラムMYNAMEを作成してください。【例題96】に続けるか、あるいは新しいファイルにファイルFUNCSをロードして、そのファイルにサブプログラムMYNAMEをセーブしてください。

解答例

```
GRPH, F, N　(AXパソコン　Alt, F, N)
GRPH, F, L, FUNCS　(AXパソコン　Alt, F, L, FUNCS)

SUB　MYNAME　(S$)
      PRINT　"私の名前は　";S$
END　SUB
GRPH　　　　(AXパソコン　Alt)
F
S
```

解説　**■パラメータ**

サブプログラムに渡るパラメータが文字列ですので、サブプログラム側では文字列型の変数($付き)で定義してください。

サブファイルの常駐

作成した関数やサブプログラムを呼び出して使うためには、あらかじめその関数やサブプログラムをメインモジュールへ読み込んで常駐させておく必要があります。これを、サブファイルの常駐といいます。サブファイルの常駐は、ファイルメニューの中から次のように選択実行します。

サブファイルの常駐

PC-9801	AX パソコン
[GRPH], [F], [L]　ファイル名指定	[Alt], [F], [L]　ファイル名指定

サブファイルを常駐させることによって、その中に書かれている関数やサブプログラムが使えるようになります。サブファイルを常駐させると、その中の使用可能な関数がDECLARE文とともに表示されます。サブプログラムについては、常駐の時点では表示されませんが、使用可能です。

【例題98】　サブファイルの常駐

新しいファイルにサブファイルFUNCSを常駐させてください。

解答例

[GRPH]　（AX パソコン　[Alt]）
[F]
[N]
[GRPH]　（AX パソコン　[Alt]）
[F]
[L]
FUNCS　[↵]

実行例

```
DECLARE FUNCTION MAX!(A!, B!)
```

解説 ■DECLARE FUNCTION MAX!(A!, B!)

サブファイルが常駐すると、サブファイルの使用宣言文 DECLARE が自動的に新しいファイルに書き込まれます。

■単精度浮動小数点

関数 MAX の定義のときには入力しなかった！が、DECLARE 文では表れています。これは、変数 A, B と関数 MAX がそれぞれデフォルトの単精度小数点数の範囲で有効なことを示しています。

■DECLARE SUB ?

ただし、DECLARE SUB はそのサブプログラムが呼び出されて実行された後の保存の段階で画面に表示されます。

変数名の重複

関数やサブプログラムの定義の中で使う関数と同じ名前の変数が、メインプログラムや他の関数やサブプログラムの定義の中の変数と重複した場合はどうなるのでしょうか？

安心してください。Quick BASIC では、関数やサブプログラムの定義の中で使われる関数は、その中でしか有効ではありません。関数やサブプログラムの外とは一切関係がありません。このように、特定の範囲に限って有効な変数をローカル変数といいます。

プロシージャをたくさん作って、プログラムの開発効率をアップしましょう。

プロシージャの呼び出し

常駐しているサブファイルに書かれているプロシージャ（関数，サブプログラム）は、その名前に続けて( )内にパラメータを入れて呼び出します。ただし、サブプログラムの呼び出しには、CALL 文が必要です。

プロシージャの呼び出し

関数	関数名(パラメータ，…)	例　MAX(12, 25)
サブプログラム	CALL　サブプログラム(パラメータ)	例　CALL　MYNAME("山田")

【例題99】　プロシージャの呼び出し

サブファイルの常駐しているファイルに12と25の大きい方をプリントする命令と、自分の名前を書く命令を追加し実行してください。

解答例

```
DECLARE  FUNCTION  MAX!(A!, B!)
PRINT  MAX(12, 25)
CALL  MYNAME("山田")
```

実行例

```
25
私の名前は 山田
```

解説　■CALL…( )

サブプログラムを呼び出す場合は、CALL 文が必要です。関数では必要ありません。

メインモジュールの保存

新しいファイルにサブファイルを常駐させると、ファイル名はサブファイルのものになってしまいます。サブファイルを常駐させたメインファイル(メインモジュール)は、必ず「名前を変えて保存」してください。「名前を変えてセーブ」は、ファイルメニューの中にあります。

名前を変えて保存

PC-9801

GRPH, F, A 新しい名前 ⏎

AX パソコン

Alt, F, A 新しい名前 ⏎

【例題100】 メインモジュールの保存

サブファイル FUNCS が常駐しているメインモジュールを、MYMAIN という名のファイルに保存してください。

解答例

PC-9801

GRPH, F, A MYMAIN ⏎

AX パソコン

Alt, F, A MYMAIN ⏎

解説

■**名前を変えて**

サブファイルは部品集、メインファイルはそれを使った全体プログラムという考え方をテッテイするために、メインモジュールとサブファイルは別の名前でセーブすることをすすめます。

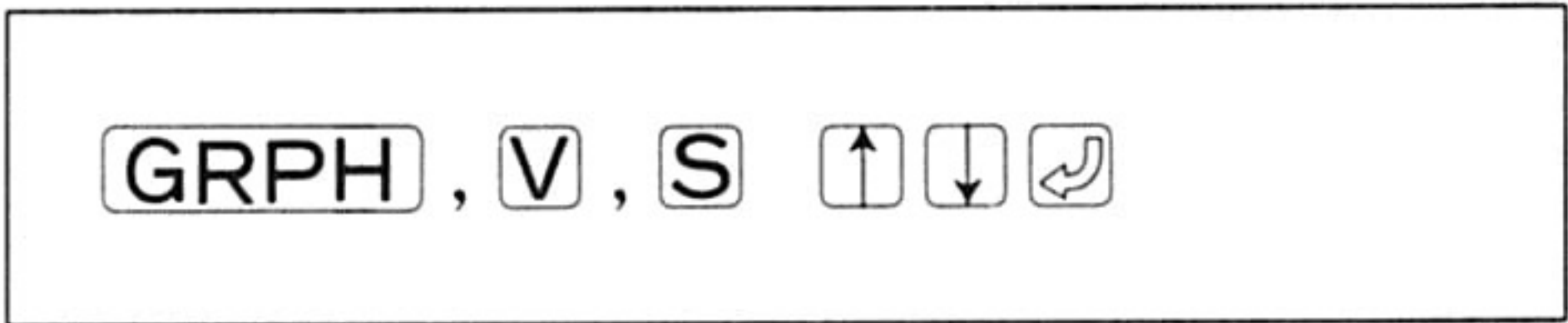
■**DECLARE SUB…**

セーブしたあとの画面を見てください。DECLARE SUB… が表れています。

プロシージャの編集

プロシージャは、そのプロシージャが常駐しているメインモジュールから直接編集できます。編集対象のプロシージャのソースコードは、次のようにして選択できます。

プロシージャの選択

GRPH , V , S　↑↓⏎

プロシージャからメインモジュールに戻る場合も、同様にしてメインモジュール名を選択してください。

【例題101】　プロシージャの編集

自分の名前をプリントするサブプログラムMYNAMEの内容を修正してMY　NAME ISの見出しに変更してください。

解答例

GRPH (AXパソコン Alt) V S MYNAME ⏎	SUB MYNAME(S$) PRINT "MY NAME IS ";S$ END SUB

解説

■切り替え

別のプロシージャを見たりメインモジュールに戻る場合にも、GRAPH (Alt, V, S) でモジュールを選択してください。

■サブファイルはもとのまま

上の例のように、常駐しているメインプログラムからプロシージャを修正した場合は、サブファイル内のプロシージャは修正されません。新しいファイルに常駐させる形で呼び出してみてください。変更されていません。

■再常駐

逆にサブファイル内のプロシージャを修正した場合は、そのプロシージャを使っているメインモジュール側では、再度サブファイル常駐の手続きが必要です。

プロシージャと変数

ローカル変数

少しむずかしい話になりますが、プロシージャの解説には欠かせないことなので、本章の最後に「ローカル変数」と「参照による呼び出し」について解説します。プロシージャについて使いこなしてくると、気になる話題です。最初は読み飛ばしてもかまいません。

プロシージャの定義側で使う変数、例えば

```
FUNCTION  MAX(A, B)
     ⋮
END  FUNCTION
```

の A, B といった変数を、仮引数(かりひきすう)といいます。この仮引数は、メインプログラムとは一切関係のないプロシージャの中でのみ有効な関数です。同じ名前の変数が、メインプログラムや他のプロシージャにあってもかまいません。プロシージャの中でのみ有効な変数を、ローカル変数といいます。

プロシージャを作成するときに、仮引数の名前を他のことは気にせずに命名できることは、プログラムをグループで開発するときに大いに力を発揮します。

参照による呼び出し

メインプログラム、すなわちプロシージャを呼び出す側で関数やサブプログラムにパラメータとして記述する変数を実引数(じつひきすう)といいます。この実引数については、その値だけでなく変数を直接参照できる形式(参照渡し)で渡されます。したがって、プロシージャ側で仮引数の値に加工をすれば、そのままメイン側での実引数の値が加工されることになります。この点は、C言語などの値による呼び出しと異なる点です。

第2章　ファイル

第２章では、キーボードから入力したデータをファイルに書き込んだり、ファイルの中のデータを読み取って画面に表示したりするファイルとの入出力の方法を学習しましょう。

第２章の概要

■ファイルの種類

ファイルには、次の２つがあります。

(1) テキストファイル

(2) バイナリファイル

■ファイルの構成

ファイルをその中のデータ構造から分けると、次の２つに分類できます。

(1) シーケンシャルファイル

(2) ランダムアクセスファイル

■ファイル操作の一般的規則

ファイルを操作するためには、ファイルをオープンする、ファイルをクローズするという儀式が必要です。ファイルは、ファイル番号で扱われます。

■シーケンシャルファイル

○ オープン　OPEN　　○ 書き込み　PRINT #, WRITE #

○ クローズ　CLOSE　　○ 読み込み　INPUT #

■ファイルの終端　EOF( )

■複数のファイル

■ランダムアクセスファイル

ランダムアクセスファイルとの入出力は、レコードを介して行ないます。

■レコードの定義　TYPE ～ END TYPE

レコードは、一定の長さの複数データからなる構造を持っています。

■レコードの確保　DIM…AS…

■レコードのメンバー

■ランダムアクセスファイル

○ オープンとクローズ　OPEN, CLOSE　　○ ファイルへの書き込み　PUT

○ 読み書きの単位　レコード　　○ ファイルからの読み込み　GET

■レコードの指定

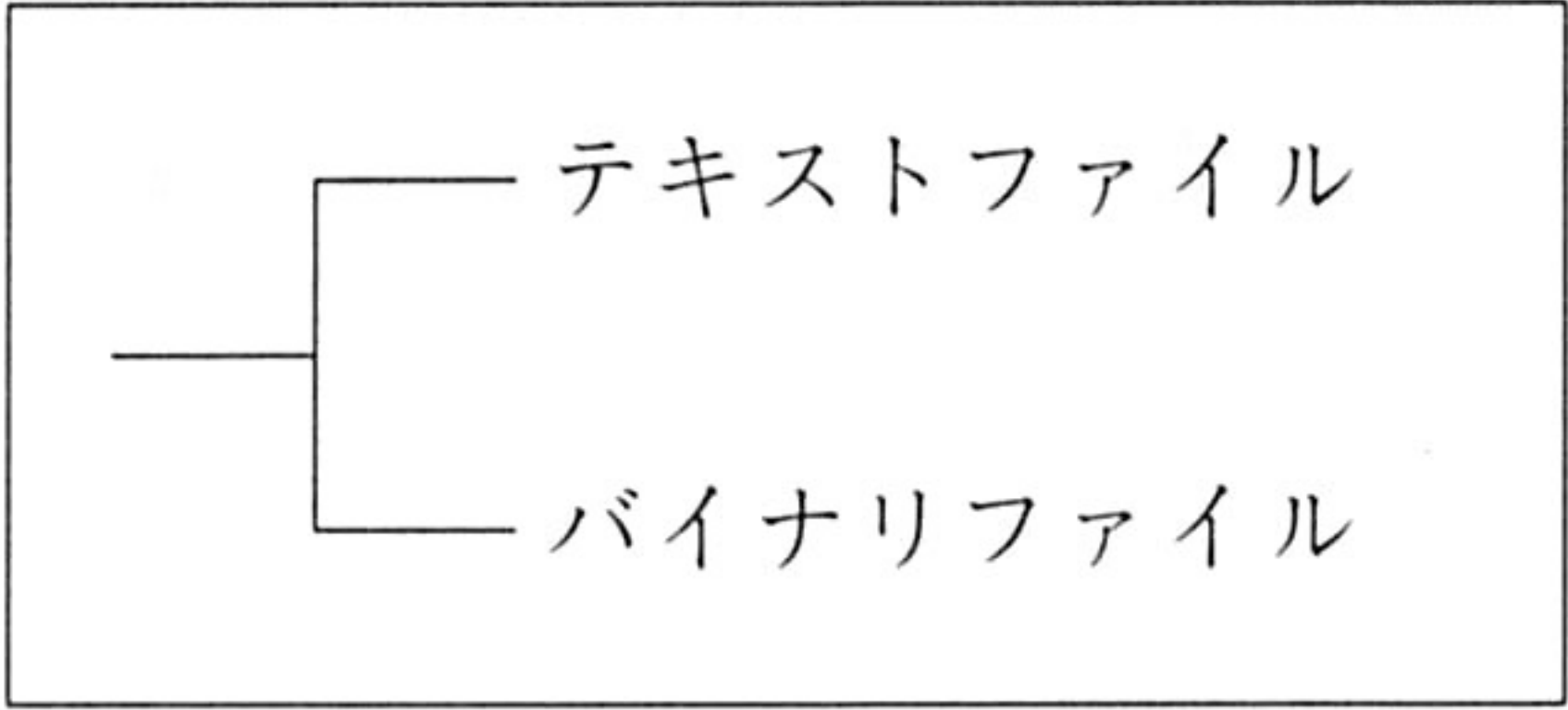
ファイルの種類

テキストファイル・バイナリファイル

ファイルには、その内容が文として意味のある(解読可能な)テキスト型ファイルと、主としてコンピュータへの命令からなり我々には直接解読できないバイナリファイルとがあります。

Quick BASIC でプログラムをセーブするときにできる拡張子 BAS や、EXEファイル実行によってできる拡張子 EXE のファイルは、バイナリファイルの例です。住所録の人名や住所などのデータファイルはテキストファイルになります。

ファイル

```
      ┌── テキストファイル
 ─────┤
      └── バイナリファイル
```

本章では、テキストファイルへのデータの書き込みや読み込みなどを学習します。

MS-DOS とテキストファイル

テキストファイルを操作する MS-DOS のコマンドを整理しておきます。

- テキストファイルの読み取り　TYPE　ファイル名 ↵
- テキストファイルの印字　TYPE　ファイル名 ＞ PRN ↵
- テキストファイルの作成(間便法)

　COPY　CON　ファイル名 ↵
　　キーボード入力
　　　⋮
　　CTRL ＋ Z ↵
　TYPE　CON ＞ ファイル名 ↵
　　<同上>

数行程度のテキストファイルなら、上の間便法で作成するのが一番簡単です。

ファイルの構成

ファイルに書かれているデータの構成によって、ファイルは次の2とおりに分類できます。

(1) シーケンシャルファイル………ファイルの先頭から連続して書き込まれているファイル

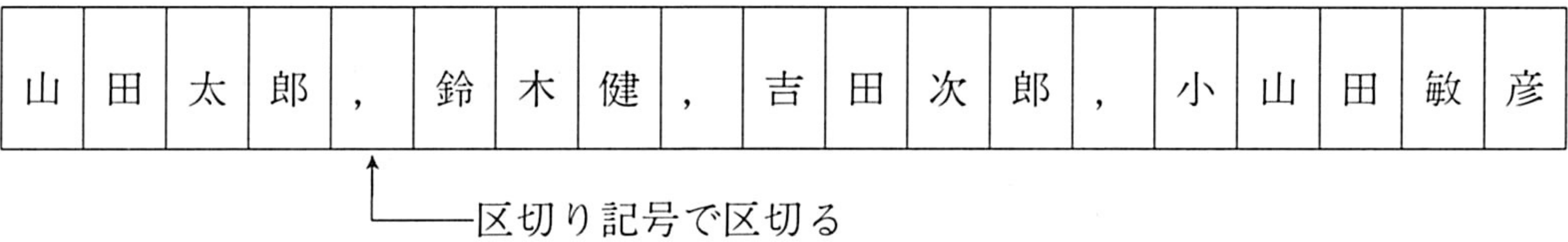
(2) ランダムアクセスファイル……ファイルの中を区切って一定の長さごとに書き込まれているファイル

あまり正しい例ではありませんが、シーケンシャルファイルはカセットテープみたいなものです。3曲目が「百万本のバラ」と分かっていても、その曲を聴くためには1曲目から聴くか、聴き飛ばしながら頭を出すことになります。一方、ランダムアクセスファイルはＬＰレコードのようなものです。3曲目の「百万本のバラ」は、その溝の位置へ針を置くことで即座に聴き始めることができます。

シーケンシャルファイル

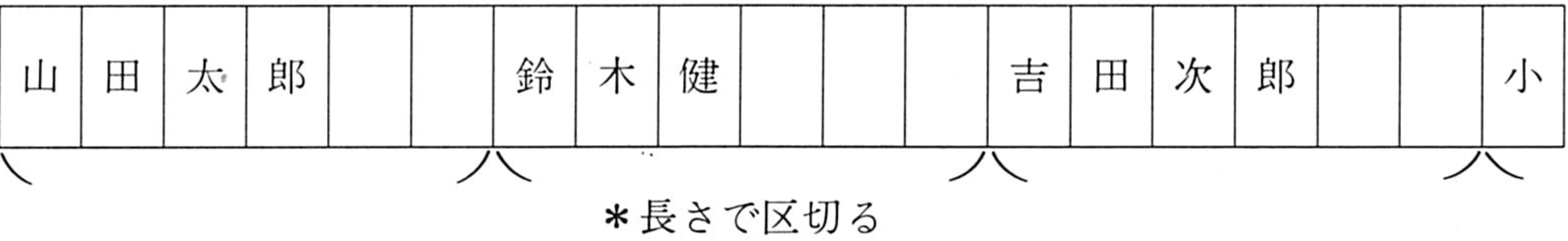
シーケンシャルファイルは、長さを決めずデータのままにセーブされているファイルともいえます。各データの区切りには、区切り記号が必要です。

山	田	太	郎	,	鈴	木	健	,	吉	田	次	郎	,	小	山	田	敏	彦

↑区切り記号で区切る

ランダムアクセスファイル

ランダムアクセスファイルは、長さを決めてデータを固定的な長さにしてセーブするファイルともいえます。

山	田	太	郎			鈴	木	健				吉	田	次	郎			小

＊長さで区切る

ランダムアクセスファイルは無駄なスペースが入るために収納効率は悪いのですが、N番目のデータは先頭からN＊(1個のデータ長)の位置にあることが計算から分かるので、高速のデータの検索には便利です。

第2章の前半ではシーケンシャルファイルを、後半ではランダムアクセスファイルを学習します。

ファイル操作の一般的規則

オープンとクローズ

シーケンシャルファイルにしろランダムアクセスファイルにしろ、ファイルを操作する場合に必要な手続きが2つあります。それは、使うファイルをオープンするということと、使い終わったファイルはクローズするという2点です。皆さんもノートに書くときには、ノートをオープンし書き終わったらクローズしますね。同じことです。

OPEN と CLOSE

ファイルをオープンするコマンドはOPEN、クローズするコマンドはCLOSEです。対象となるファイルがシーケンシャルファイルかランダムアクセスファイルかによって、OPENコマンドの使い方が少し変わります。

ファイル番号

MS-DOSでは、同時にいくつかのファイルをオープンすることができます。オープンしているファイルは、OPENコマンドで指定したファイル番号で管理されます。

書き込みと読み込み

データをディスクに書くことを書き込み、あるいはファイルへの出力といいます。ディスク上のデータをメモリ空間へ読むことを読み込み、あるいはファイルからの入力といいます。

ファイル内の位置

ファイルとデータの入出力、すなわちファイルにデータを書いたり、ファイルからデータを読んだりするときはいつも、「いまファイルのどこにいるのか」ということを意識してください。

ファイルをオープンした直後は、ファイルの先頭にいます。ひとつ読み書きするたびに、ひとつ分先へ進みます。

シーケンシャルファイル　オープンとクローズ　OPEN, CLOSE

オープン

シーケンシャルファイルをオープンする場合は、そのオープン後に行なう仕事によって、コマンドOPENの使い方が異なってきます。

```
オープンしたあとファイルの先頭から書き込む
 OPEN "ファイル名" FOR OUTPUT AS ファイル番号
  (例) OPEN "TEST.SQF" FOR OUTPUT AS ♯1
       CLOSE ♯1
オープンしたあとファイルの末尾に追加で書き込む
 OPEN "ファイル名" FOR APPEND AS ファイル番号
  (例) OPEN "TEST.SQF" FOR APPEND AS ♯1
       CLOSE ♯1
オープンしたあとファイルの先頭から読む
 OPEN "ファイル名" FOR INPUT AS ファイル番号
  (例) OPEN "TEST.SQF" FOR INPUT AS ♯1
       CLOSE ♯1
```

ただし、読み込みのためにオープンする(FOR INPUT)場合は、そのファイルがすでに存在していなくてはなりません。存在しないファイルからデータを読み取ることはできません。

クローズ

ファイルをクローズするコマンドは、CLOSEです。

ファイルのクローズ

```
CLOSE ファイル番号
```

(例)　CLOSE ♯1

ファイル番号

ファイル番号には1からの整数を使いますが、他の数値データと区分するために、♯1のように♯を付けます。

シーケンシャルファイル 書き込み PRINT #, WRITE

シーケンシャルファイルにデータを書き込むコマンドには、2つあります。

シーケンシャルファイルへの書き込み

```
PRINT  ファイル番号, データ
WRITE  ファイル番号, データ
```

PRINT コマンドによるファイルへの書き込みでは、ファイル番号指定がなければ画面に表示されるであろう表示そのものが、ひとつのデータとしてファイルに書き込まれます。

(例)
```
OPEN "TEST.SQF" FOR OUTPUT AS #1
PRINT #1, "ABC", "DEF"; "GHI"
CLOSE #1
```

```
ABC            DEFGHI
```

ファイルの中のデータ

WRITE コマンドによるファイルへの書き込みでは、命令文のデータそのものがファイルに書き込まれます。ただし、セミコロンはカンマに置き換わります。

(例)
```
OPEN "TEST.SQF" FOR OUTPUT AS #1
WRITE #1, "ABC", "DEF"; "GHI"
CLOSE #1
```

```
"ABC", "DEF", "GHI"
```

ファイルの中のデータ

【例題102】　シーケンシャルファイル(書き込み)　PRINT ＃

シーケンシャルファイル"TEST1.SQF"を新たに作成して、次のように3つのデータをPRINTコマンドで書き込んでください。

ABC
DEF
GHI

解答例

```
OPEN "TEST1.SQF" FOR OUTPUT AS #1
PRINT #1,"ABC"
PRINT #1,"DEF"
PRINT #1,"GHI"
CLOSE #1
```

解説

■確認

上のプログラムを SHIFT ＋ f·5 で実行してください。ファイルTEST.SQFに正しくデータが書き込まれているか確認しておきましょう。Quick BSSICを一時的に抜けて、MS-DOSのコマンドTYPEで確認します。

```
GRPH    (AXパソコン  Alt)
F
D
A>TYPE TEST1.SQF ⏎
ABC
DEF
GHI
A>EXIT ⏎
```

【例題103】 シーケンシャルファイル(書き込み) WRITE #

シーケンシャルファイル"TEST2.SQF"を新たに作成し、WRITEコマンドを使って次のようにデータをファイルに書き込んでください。
"ABC", "DEF", "GHI"

解答例

```
OPEN "TEST2.SQF" FOR OUTPUT AS #1
WRITE #1, "ABC", "DEF", "GHI"
CLOSE #1
```

解説 **■確認**

前問と同じように、MS-DOSのTYPEコマンドでファイルの内容を確認してください。

```
[GRPH]  (AXパソコン [Alt])
[F]
[D]
A>TYPE TEST2.SQF [↵]
"ABC", "DEF", "GHI"
A>EXIT [↵]
```

シーケンシャルファイル　読み込み　INPUT ＃

すでにデータの書かれているシーケンシャルファイルからデータを読むコマンドは、INPUTです。これは、データの書き込みにPRINTコマンド、WRITEコマンドのいずれを使ったかに関係ありません。

シーケンシャルファイルからの読み込み

```
INPUT　ファイル番号, 受け取る変数
```

【例題104】　シーケンシャルファイル(読み込み)　INPUT ＃

ファイルTEST1.SQFから最初のデータをひとつ読み取って表示してください。

解答例

```
OPEN "TEST1.SQF" FOR INPUT AS #1
INPUT #1, A$
CLOSE #1
PRINT A$
```

実行例

```
ABC
```

解説　**■A$で受け取る**

上の解答例では、A$という文字型の変数で受け取っていますが、これをもしAという数値型の変数で受け取るとどうなるでしょうか。やってみてください。0と表示されると思いますが、これは、ABCというデータをその内容が値0の数値として読み取ったからです。

【例題105】 シーケンシャルファイル(読み込み) INPUT

ファイル TEST2.SQF から最初のデータをひとつ読み取って表示してください。

解答例

```
OPEN "TEST2.SQF" FOR INPUT AS #1
INPUT #1, A$
CLOSE #1
PRINT A$
```

実行例

```
ABC
```

解説 ■**引用符 " "**

引用符の付いているデータ "ABC" が、文字型変数 A$ に入力されます。

PRINT は、" " を表示しません。

■**データの区切り ,**

INPUT 文は、カンマをデータの区切りとして認識します。

標準的なシーケンシャルファイル

データファイルとしてのシーケンシャルファイルは、次のように文字データは引用符でくくり、数値データはそのままにして、データの区切りはカンマとするものです。

"YAMADA", "YOKOHAMA", 24, "0466-22-…"

市販されているソフトウェア(ワープロやデータベース)では、この形式のシーケンシャルファイルを取り込むことができるようになっているのが普通です。

ファイルの終端　EOF()

ファイルの終端は、EOF()関数によって検出することができます。EOFは、End Of Fileの意味です。

ファイルの終端

```
EOF(ファイル番号)        ファイルの終端では真 ─┐
                                                ├ を返す
                         ファイルの途中では偽 ─┘

                 (注)  このファイル番号は、#なしで指定します。
```

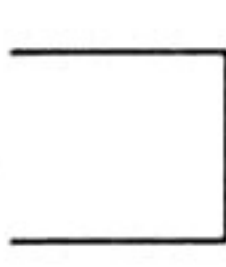
このEOF()関数を使いファイルの最初から最後まで一定の動作を繰り返すプログラムは次の形式になります。

```
DO  UNTIL  EOF(ファイル番号)
      …
      …
LOOP
```

【例題106】　ファイルの終端　EOF()

シーケンシャルファイルTEST2.SQFから、全データを読み取り表示してください。

解答例

```
OPEN "TEST2.SQF" FOR INPUT AS #1
DO UNTIL EOF(1)
    INPUT #1, A$
    PRINT A$
LOOP
CLOSE #1
```

複数のファイル

MS-DOS では、複数のファイルを同時にオープンして操作することができます。ここでは、複数ファイルの操作の例として、ファイルのコピーをしてみましょう。

ファイルをコピーする手順は次のとおりです。

(1)コピー元のファイルを読み込み(FOR　INPUT)でオープンする
(2)コピー先のファイルを書き込み(FOR　OUTPUT)でオープンする
(3)コピー元ファイルからデータを読む ─┐コピー元ファイルの終端
(4)コピー先ファイルへそのデータを書く ─┘まで繰り返す
(5)2つのファイルをクローズする

【例題107】　ファイルの複写

ファイル TEST2.SQF を TEST3.SQF にコピーしてください。

解答例

```
OPEN "TEST2.SQF" FOR INPUT AS #1
OPEN "TEST3.SQF" FOR OUTPUT AS #2
DO UNTIL EOF(1)
     INPUT #1, A$
     WRITE #2, A$
LOOP
CLOSE #1
CLOSE #2
```

解説　**■カンマ→改行**

上の解答例では、複写元のファイルのデータ区切り記号カンマが、複写先のファイルでは ↵ に置き換わっていますので、厳密な複写にはなっていません。

■確認

ファイルが複写されたかどうかは、MS-DOS の TYPE コマンドで確認してください。

ランダムアクセスファイル

固定長

本章のこのセクション以降では、ランダムアクセスファイルを学習します。ランダムアクセスファイルは、個々のデータを一定の長さで格納したファイルのことです。

ファイルへの入出力に際して、データを一定の長さに整形する必要があります。あるいは、一定の長さのものに入れて、それごとファイルに書き込んでしまうといった方が分かりやすいかもしれません。いずれにしろ、一定の長さを持った容器が必要になります。

レコード

ランダムアクセスファイルへの入出力は、レコードと呼ばれる、ユーザがその扱うデータに合わせて作成(定義と宣言)する複合構造の変数を使って行なわれます。

ランダムアクセスファイルとの橋渡しをするこのレコードのことを、バッファと呼ぶこともあります。

ランダムアクセスファイルと数値データ

ランダムアクセスファイルはテキストファイルのひとつとして説明してきましたが、Quick BASIC のランダムアクセスファイルは厳密にはテキストでない部分を含んでいます。

数値データは、一定の規則にしたがって圧縮されて保存されます。整数は２バイト(アルファベット２文字)に、単精度浮動小数点は４バイトに…などと、数値は変形されて保存されます。

したがって、Quick BASIC のランダムアクセスファイルを MS-DOS の TYPE コマンドで確認すると、その一部が文字バケして正しいものは読み取ることができません。ただし、内容は OK ですから安心してください。

レコードの定義　TYPE ～ END TYPE

コンピュータで処理するデータが、複数の項目からなっていることがあります。例えば、住所録では以下の5項目をひとつのデータとして扱うのが普通でしょう。

```
人名
郵便番号
住所(1)
住所(2)
電話番号
```

Quick BASIC では、対象となるデータが複数の項目からなるレコードというものを定義できます。レコードを定義するコマンドは、TYPE ～ END TYPE です。

レコード構成の定義

```
TYPE レコード名
     変数名 AS 形式
       …       …
       …       …
END TYPE
```

上の住所録のためのレコード PERSON は、次のように定義されます。

```
TYPE PERSON
     NAMAE AS STRING * 16
     ZIP AS STRING * 8
     ADDR1 AS STRING * 26
     ADDR2 AS STRING * 26
     TEL AS STRING * 20
END TYPE
```

レコード名

左ページの例で、レコード名 PERSON はレコードの型に付けられた名前です。データ構成の型枠に付けられた名前で、変数名ではありません。

AS 形式

AS… は、その変数の型式を指定します。

```
AS  STRING * N      Nバイトの文字列
AS  INTEGER         整数
```

【例題108】　レコードの定義　TYPE ～ END TYPE

生徒ひとりひとりについて、名前と主要5教科の点数を管理するレコードを考えてください。

解答例

```
TYPE SEITO
    NAMAE AS STRING * 16
    KOKUGO AS INTEGER
    SUUGAKU AS INTEGER
    EIGO AS INTEGER
    RIKA AS INTEGER
    SYAKAI AS INTEGER
END TYPE
```

レコードの確保　DIM…AS…

レコードの定義で行なわれるのは、レコードの構成の指示だけです。実際にそのレコードの形式を持った変数を確保するには、DIM 宣言子を使います。

レコードの確保

```
DIM　変数名　AS　レコード名
```

住所録などのように、同じレコードをいくつも確保する場合は、その変数を配列として宣言します。

レコード配列の確保

```
DIM　変数名(要素数)　AS　レコード名
```

住所録用に50人分のレコードを確保するには、次のようにします。

```
DIM　FRIENDS(50)　AS　PERSON
```

【例題109】　レコードの確保　DIM…AS…

【例題108】で考えた生徒成績管理用のレコード SEITO を、変数名 STUDENTS で生徒の人数分(100人)確保してください。

解答例

```
DIM　STUDENTS(100)　AS　SEITO
```

レコードのメンバー　.

メンバー　.

レコードを構成するひとつひとつの項目を、レコードのメンバーといいます。レコードのメンバーは、次のようにして参照することができます。

メンバーの参照

```
変数名.メンバー名
```

【例題110】、レコードのメンバー　.

次のようなレコードを定義し、50人分の配列 FRIENDS を確保してください。
次に、その1番目のレコードにデータを入力し、表示するプログラムを作成してください。

レコード	PERSON		
メンバー	名前	NAME	長さ16バイト
	郵便番号	ZIP	長さ 8バイト
	住所(1)	ADDR1	長さ26バイト
	住所(2)	ADDR2	長さ26バイト
	電話番号	TEL	長さ20バイト

解答例

```
TYPE PERSON
    NAMAE AS STRING * 16
    ZIP AS STRING * 8
    ADDR1 AS STRING * 26
    ADDR2 AS STRING * 26
    TEL AS STRING * 20
END TYPE

DIM FRIENDS(50) AS PERSON
```

```
I = 1
INPUT "名前", FRIENDS(I).NAMAE
INPUT "郵便番号", FRIENDS(I).ZIP
INPUT "住所(1)", FRIENDS(I).ADDR1
INPUT "住所(2)", FRIENDS(I).ADDR2
INPUT "電話番号", FRIENDS(I).TEL
CLS
PRINT FRIENDS(I).NAMAE
PRINT FRIENDS(I).ZIP
PRINT FRIENDS(I).ADDR1
PRINT FRIENDS(I).ADDR2
PRINT FRIENDS(I).TEL
```

【例題111】 複数のレコード

【例題110】を名前の入力に対して単に⏎だけが押されたときに終了するようにして、複数人まとめて登録できるようにしてください。

解答例

```
TYPE PERSON
    NAMAE AS STRING * 16
    ZIP AS STRING * 8
    ADDR1 AS STRING * 26
    ADDR2 AS STRING * 26
    TEL AS STRING * 20
END TYPE

DIM FRIENDS(50) AS PERSON

TRUE = 1
I = 1
```

```
DO
    INPUT "名前", FRIENDS(I).NAMAE
    IF FRIENDS(I).NAMAE = SPACE$(16) THEN
        EXIT DO
    END IF
    INPUT "郵便番号", FRIENDS(I).ZIP
    INPUT "住所1", FRIENDS(I).ADDR1
    INPUT "住所2", FRIENDS(I).ADDR2
    INPUT "電話番号", FRIENDS(I).TEL
I = I + 1
LOOP WHILE TRUE
FOR I = 1 TO 50
    IF FRIENDS(I).NAMAE = SPACE$(16) THEN
        EXIT FOR
    END IF
    PRINT FRIENDS(I).NAMAE;FRIENDS(I).ZIP;FRIENDS(I).ADDR1;
    PRINT FRIENDS(I).ADDR2;FRIENDS(I).TEL
NEXT I
```

解説　**■SPACE$(16)**

レコードPERSONの中で、NAMAEは16文字分の長さで定義されています。NAMAEは、入力が単に↵のみであったときは空文字" "ではなく、16個のスペースになります。

ランダムアクセスファイル　オープンとクローズ　OPEN, CLOSE

ランダムアクセスファイルは、一定の長さごとに区切られてデータが書き込まれているファイルです。このランダムアクセスファイルをオープンするコマンドはOPENです。

ランダムアクセスファイルのオープン

```
OPEN "ファイル名" FOR RANDOM AS ファイル番号 LEN = データの長さ
```

シーケンシャルファイルと異なって、ランダムアクセスファイルではオープンに際して入力用、出力用の区別がありません。オープンしていれば、読み込みも書き込みもできます。

データの長さ

ランダムアクセスファイルは、オープンするときにひとつひとつのデータの長さをLEN = で指定します。この長さ指定により、ファイルの1回の読み込み量、書き込み量が決定されます。

ファイルのクローズ

ランダムアクセスファイルをクローズするコマンドは、CLOSEです。

ランダムアクセスファイルのクローズ

```
CLOSE ファイル番号
```

ランダムアクセスファイル　読み書きの単位　レコード

レコード

ランダムアクセスファイルとの入出力、すなわちランダムアクセスファイルからデータを読む、あるいはランダムアクセスファイルにデータを書く場合は、必ずレコードを使ってください。

（例）

```
TYPE PERSON
     NAMAE AS STRING * 16
     ADDR AS STRING * 26
     AGE AS INTEGER
END TYPE
DIM FRIEND AS PERSON
FRIEND.NAMAE = "山田 太郎"
FRIEND.ADDR = "横浜市"
FRIEND.AGE = 18
OPEN "TEST.RAF" FOR RANDOM AS #1 LEN = LEN(FRIEND)
PUT #1,,FRIEND
CLOSE #1
```

上の（例）のように、必ずレコードにデータをセットして書き込みや読み込みを行ないます。

PUTについては、次のセクションで解説します。

ランダムアクセスファイル　ファイルへの書き込み　PUT

ランダムアクセスファイルにデータ(レコード)を書き込むコマンドはPUTです。

ランダムアクセスファイルへの書き込み

```
PUT　ファイル番号, 位置, レコード
```

位置

ランダムアクセスファイルは、データを書き込む位置(先頭から第何番目のレコードというように)を指定できます。OPENした直後に、この位置指定を省略してPUT文を使うと、ファイルの先頭から書き込みが行なわれます。

【例題112】　ランダムアクセスファイル(書き込み)　PUT

人名(16文字), 住所(26文字), 年齢(整数)からなるレコードPERSONを定義し、その型の変数FRIENDにデータをセットしたあと、ランダムアクセスファイルTEST.RAFに保存してください。

解答例

```
TYPE PERSON
    NAMAE AS STRING * 16
    ADDR AS STRING * 26
    AGE AS INTEGER
END TYPE
DIM FRIEND AS PERSON
FRIEND.NAMAE = "山田 太郎"
FRIEND.ADDR = "横浜市"
FRIEND.AGE = 18
OPEN "TEST.RAF" FOR RANDOM AS #1 LEN = LEN(FRIEND)
PUT #1,,FRIEND
CLOSE #1
```

ランダムアクセスファイル　ファイルからの読み込み　GET

ランダムアクセスファイルに書き込まれたデータを読み込むコマンドはGETです。

ランダムアクセスファイルの読み込み

```
GET　ファイル番号, 位置, レコード
```

位置

ランダムアクセスファイルでは、データを読み込む位置（先頭から第何番目のレコードというように）を指定することができます。OPENした直後に、この位置指定を省略してGET文を使うと、ファイルの先頭から読み込みが行なわれます。

【例題113】　ランダムアクセスファイル（読み込み）　GET

ランダムアクセスファイルTEST.RAFからデータを読み込んで表示してください。

解答例

```
TYPE PERSON
    NAMAE A$ STRING * 16
    ADDR AS STRING * 26
    AGE AS INTEGER
END TYPE
DIM FRIEND AS PERSON
OPEN "TEST.RAF" FOR RANDOM AS #1 LEN = LEN(FRIEND)
GET #1,,FRIEND
CLOSE #1
PRINT FRIEND.NAMAE;FRIEND.ADDR;FRIEND.AGE
```

レコードの指定

ランダムアクセスファイルでは、第何番目のレコードとしてデータを書き込むか、あるいは読み込むかというレコードの位置を指定できます。PUT や GET の第2番目のパラメータとして、レコードの位置(レコード番号)を指定してください。

【例題114】 レコードの指定

【例題112】で作成した TEST.RAF に、次のデータを追加してください。追加後、第2番目のレコードを読み込み表示してください。

名前 "鈴木 二郎"
住所 "藤沢市"
年齢 24

解答例

```
TYPE PERSON
    NAMAE AS STRING * 16
    ADDR AS STRING * 26
    AGE AS INTEGER
END TYPE
DIM FRIENDI AS PERSON
DIM FRIENDO AS PERSON
FRIENDI.NAMAE = "鈴木 二郎"
FRIENDI.ADDR = "藤沢市"
FRIENDI.AGE = 24
OPEN "TEST.RAF" FOR RAODOM AS #1 LEN = LEN(FRIENDI)
PUT #1, 2, FRIENDI
GET #1, 2, FRIENDO
CLOSE #1
PRINT FRIENDO.NAMAE;FRIENDO.ADDR;FRIENDO.AGE
```

第3章　グラフィックⅡ

グラフィックについては、第2部の第4章で簡単に扱いましたが、本章では、グラフィックのより高度なテクニックについて学びましょう。テーマは、

(1) ベタ塗りではなく、タイルパターンで図形を塗る

(2) アニメーション

の2つです。いずれも、16進数によるパターン表現が必要です。本章の始めに、16進数の学習をします。

第3章の概要

■2進数

グラフィックの基本は、2進数を知ることから始まります。

■16進数

2進数を4個ずつまとめて16進数にすると、扱いやすくなります。

■ラインスタイル

一点鎖線や点線などは、LINE のラインスタイル指定で描きます。

■光の3原色 RGB と輝度

模様塗りや中間色は、色の組み合せで表現します。

■色を塗る　PAINT

色を塗るステートメントは PAINT です。

■タイルパターン(1), (2)

模様を表現する文字列を、タイルパターンといいます。タイルパターンは、横に8ドット、縦に64ドットまでです。

■アニメーション

アニメーション、すなわち動く絵は、描いては消すの繰り返しです。

■イメージの入出力

絵のイメージは、配列に GET して画面に PUT します。

○原画の作図

○配列への格納　GET

○配列の描き出し　PUT

○再描画による消し込み

2進数

10進数

私たちは、ふだん0～9を基調とした10進数を使っています。この10進数では、0から数えていって、9の次はその左に1を書きその桁は0に戻ります。これが10（ジュウ）です。

2進数

一方、コンピュータの世界では、1と0からだけなる2進数が基本になります。0，1と数えて、その次は左の桁に1を書いてその桁は0にします。形は1 0（イチゼロ）ですが、その値は1 0（ジュウ）ではなく「2（ニ）」です。1 0（イチゼロ）の次は1 1（イチイチ）で値は3（サン）になります。

次の表に2進数と10進数の対応を示します。

2進数	10進数
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16
⋮	⋮

16進数

16進数

コンピュータの世界での基本は2進数ですが、この2進数で少し大きな値(例えば250(ニヒャクゴジュウ))を表現するには、多くの1と0が必要になります。

値	2進表記
250	11111010

これでは不便ですから、2進数の4桁をひとつにまとめた16進数というものを使います。16進数では、0，1，2…9と数えていき、値10に相当する記号をA、値11に相当する記号をB、…値15に相当する記号をFとします。

16進数	10進数	2進数	16進数	10進数	2進数
0	0	0	10	16	10000
1	1	1	11	17	10001
2	2	10	12	18	10010
3	3	11	13	19	10011
4	4	100	14	20	10100
5	5	101	15	21	10101
6	6	110	16	22	10110
7	7	111	17	23	10111
8	8	1000	18	24	11000
9	9	1001	19	25	11001
A	10	1010	1A	26	11010
B	11	1011	1B	27	11011
C	12	1100	1C	28	11100
D	13	1101	1D	29	11101
E	14	1110	1E	30	11110
F	15	1111	1F	31	11111

&H

Quick BASIC では、2進数を直接扱いません。扱う数値は16進数か10進数です。そこで、16進数にはその頭に &H(アンパサンドエッチ)を付けることにして区別しています。16進数の &H10 の10進数での値は16(ジュウロク)です。

2進数→16進数

2進数から16進数への表現の変換は、グラフィックを利用する上で避けて通れない事項ですから少し練習をしておきましょう。

2進数を16進数に直すには、2進数を下(しも)から4桁毎に区切って、その区切った4桁の2進数を16進数で表現してください。先頭に &H を付ければ Quick BASIC で使える16進数の表現になります。

(例)

2進数　　1 1 1 1 0 1 0

下から4桁に分ける

1 1 1　　1 0 1 0

おのおのを16進数で読む

7　　A

つなげて &H をつける

16進数　　&H7A

【例題115】　2進数から16進数へ

次の2進数を16進数に直してください。

2進数　　1 1 0 1 0 1 1 0

解答例

2進数　　1 1 0 1 0 1 1 0

分ける　　1 1 0 1　　0 1 1 0

読む　　D　　5　　(答)　&HD5

ラインスタイル

ラインスタイル

LINEは線や長方形を描くコマンドですが、その最後のパラメータとしてどの様な線を描くかを指定できます。LINEが描く線の模様のことをラインスタイルといいます。

LINE

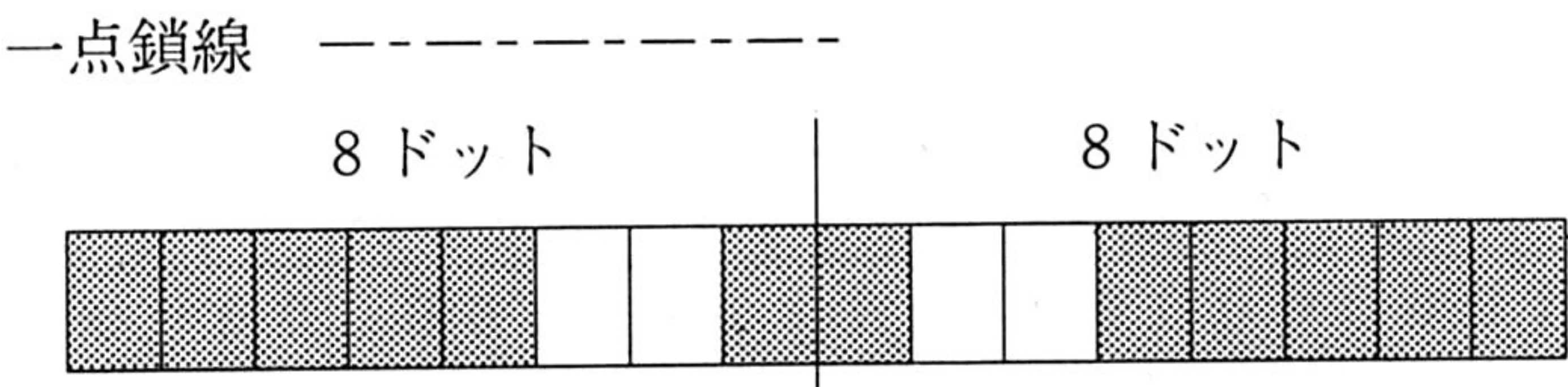
LINEの書式

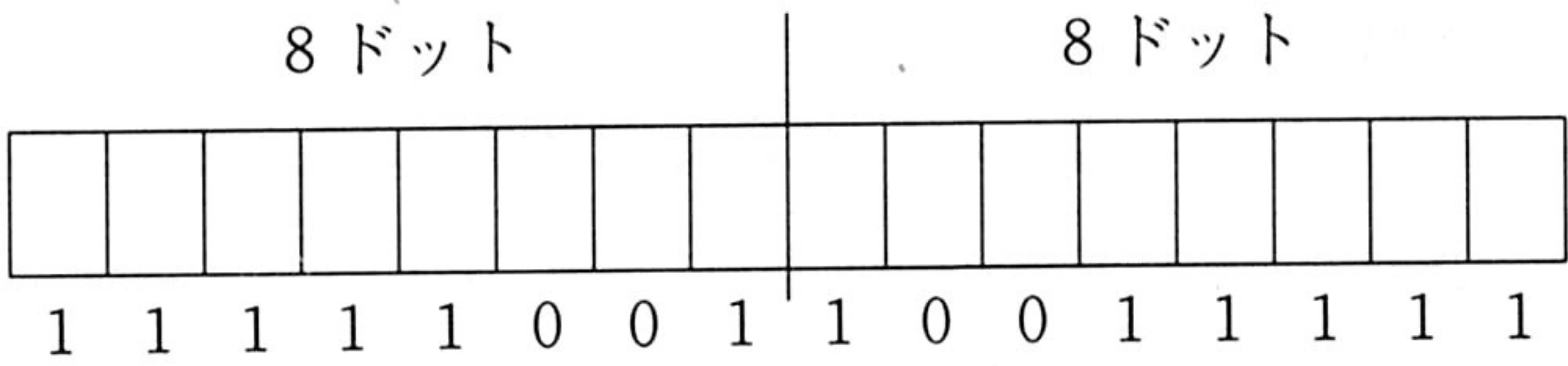
LINE(開始点) − (終了点), 色, ボックスオプション, ラインスタイル

ラインスタイルは、横に16ドットのパターンで指定します。

LINEによって次のような線を描くことを考えます。

一点鎖線　―-―-―-―-―-

8ドット　8ドット

一点鎖線はグラフィックの1ドット単位で考えると、上のように左から5ドット描いて2ドット休み、2ドット描いて2ドット休んで5ドット描けばよさそうです。光らせるドットを1、光らせないドットを0とすると、一点鎖線の一単位は、次のようになります。

8ドット　8ドット

1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1

この16個の1と0を2進数と見て16進数で表記したものが、LINEの最後のパラメータラインスタイルになります。

1111　1001　1001　1111

F　9　9　F　&HF99F

一点鎖線のラインスタイルは、&HF99Fになります。

【例題116】　ラインスタイル＜一点鎖線＞

一点鎖線を画面の中央に引いてください。色は赤です。

解答例

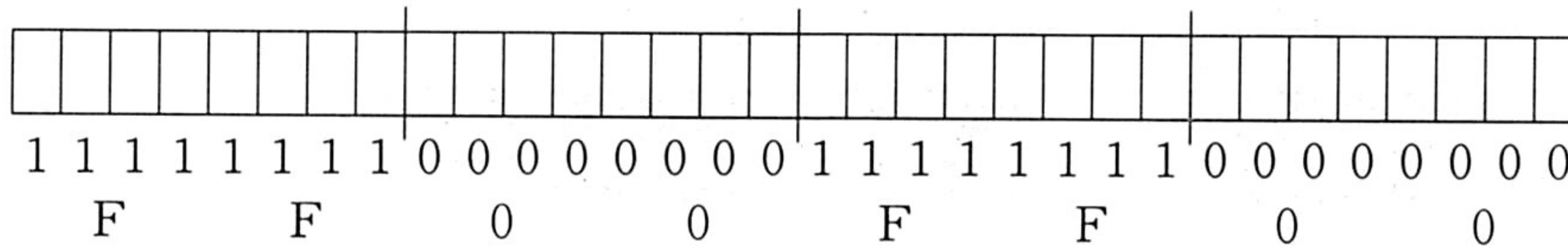
```
LINE (0, 200) - (639, 200), 4,,&HF99F
```

【例題117】　ラインスタイル＜点線＞

画面中央に点線を引いてください。色は赤です。

解答例

点線のラインスタイルを考える

1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0
F　F	0　0	F　F	0　0

プログラム

```
CLS
LINE (0, 200) - (639, 200), 4,,&HFF00
```

【例題118】　ラインスタイル＜模様＞

画面全体を赤と白の垂れ幕模様にしてください。

解答例

PC-9801

```
FOR I = 0 TO 399
   LINE (0, I) - (639, I), 4,,&HFF00
   LINE (0, I) - (639, I), 7,,&H00FF
NEXT I
```

AX パソコン

```
FOR I = 1 TO 479
     LINE (0, I) - (639, I), 4,,&HFF00
     LINE (0, I) - (639, I), 7,,&H00FF
NEXT I
```

光の3原色RGBと輝度

画面にいろいろなパターンを表現するためには、16進数によるパターン表現の他にカラーに対する基本的な知識が必要です。画面に表現されるカラーは、光の組み合せからなっています。

光の3原色

赤(Red)，緑(Green)，青(Blue) を光の3原色といいます。頭文字をとってRGBとも表現します。この3色の光でどの様な色でも表現できるからです。

光の3原色

赤(Red) 緑(Green) 青(Blue)

カラーコード

Quick　BASICでは、この光の3原色に次のカラーコードを与えています。

色	カラーコード
青	1
緑	2
赤	4

色の合成

青と緑を混ぜると水色になります。同様に青と赤では紫になります。このように原色を混ぜ合わせて派生的な色を作ることを色の合成といいます。合成された色のカラーコードは、元の色のカラーコードの和になります。

水色は青の1と緑の2の合成ですから、そのカラーコードは1＋2＝3になります。3原色から作られる8つの色とそのカラーコードの関係を次の表に示します。表の中で1はその色を使うこと、0は使わないことを意味しています。

合成とカラーコード

結果の色	3原色			カラーコード
―――	赤	緑	青	―――
黒	0	0	0	0
青	0	0	1	1
緑	0	1	0	2
水	0	1	1	3
赤	1	0	0	4
紫	1	0	1	5
黄	1	1	0	6
白	1	1	1	7

輝度

光の強さを表す輝度(Intensity)は、明(高輝度)と暗(低輝度)の2つが指定できます。0は明るい色、1は暗い色になります。

光の3原色(RGB)に、輝度(I)を加えたRGBIで16色を表現できます。

I	R	G	B
0	1	0	0

は明るい赤

I	R	G	B
1	1	1	0

は暗い水色

になります。

この4要素の組み合せ表を次のページに示します。

I	R	G	B	結果の色
0	0	0	0	黒
0	0	0	1	明るい青
0	0	1	0	明るい緑
0	0	1	1	明るい水色
0	1	0	0	明るい赤
0	1	0	1	明るい紫
0	1	1	0	明るい黄
0	1	1	1	明るい白
1	0	0	0	灰色
1	0	0	1	暗い青
1	0	1	0	暗い緑
1	0	1	1	暗い水色
1	1	0	0	暗い赤
1	1	0	1	暗い紫
1	1	1	0	暗い黄
1	1	1	1	暗い白

この後のタイルパターンの設定では、この表を左に90°回した表を使います。

B	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
G	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
R	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
I	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
結果の色	黒	明るい青	明るい緑	明るい水色	明るい赤	明るい紫	明るい黄	明るい白	灰色	暗い青	暗い緑	暗い水色	暗い赤	暗い紫	暗い黄	暗い白

この表とPAINT文を使うと、図形の内部をいろいろな色のいろいろなパターンで塗ることができます。

色を塗る　PAINT

PAINT

色を塗るステートメントとして、第2部ですでにPAINTを学習しました。その復習からはじめて、色を塗ることについてもう少し深く掘り下げてみましょう。

PAINTの基本書式は次のとおりでした。

PAINTの基本書式

```
PAINT　開始点, 塗る色, 止める色
```

【例題119】　PAINTの基本形

座標(100, 100)を中心に半径50の緑の円を描き、その中を赤く塗ってください。

解答例

```
CLS
CIRCLE(100, 100), 50, 2
PAINT(100, 100), 4, 2
```

模様塗り

図形の内部を単純な一色ではなく、模様塗り(例えば赤と白の縦縞塗り)するにはどのようにしたらよいのでしょうか。

それには、PAINTの第2の書式を使います。

PAINTの模様塗り書式

```
PAINT　開始点, 模様を表現する文字列, 止める色
```

模様を表現する文字列＝タイルパターン

模様を表現する文字列のことをタイルパターンといいます。ミシンのパターン縫いが、あらかじめ用意したパターンカードを入れればできるように、タイルパターンと呼ばれる文字列を用意しておいてPAINTに渡せば模様塗りができます。

タイルパターン（1）

図形内部を模様塗りするために用意する文字列をタイルパターンといいます。この文字列を使うと、色の付いたタイルを貼り合わせるように画面にパターンを表現できるのでタイルパターンと呼ばれます。

タイルパターンによる色塗り

```
PAINT 開始点，タイルパターン，止める色
```

タイルパターン

タイルパターンは、8ドット単位で表現した各ドットの色指定をCHR関数で文字列にしたものです。

（例）　赤と白の縦縞塗りの例

赤	赤	赤	赤	白	白	白	白

赤と白は、前々ページの表を使って原色と輝度に分解すると次のようになります。

	赤	赤	赤	赤	白	白	白	白	16進数読み
B	0	0	0	0	1	1	1	1	→ &H0F
G	0	0	0	0	1	1	1	1	→ &H0F
R	1	1	1	1	1	1	1	1	→ &HFF
I	0	0	0	0	0	0	0	0	→ &H00

この表を横に16進数で読んで文字に変換し加えたものが、模様を表す文字列、すなわちタイルパターンになります。

赤と白の縦縞模様のタイルパターンは次のようになります。

タイルパターン（赤と白のストライプ）

```
TILE$ = CHR$(&H0F) + CHR$(&H0F) + CHR$(&HFF) + CHR$(&H00)
```

【例題120】 模様塗り タイルパターン

(320, 200)に半径100の緑の円を描き、内部を赤白のストライプ塗りにしてください。

解答例

```
CLS
CIRCLE (320, 200), 100, 4
TILE$ = CHR$(&H0F) + CHR$(&H0F) + CHR$(&HFF) + CHR$(&H00)
PAINT (320, 200), TILE$, 4
```

中間色塗り

赤や青という原色だけでなく、オレンジやピンクなどという中間色で塗ることを考えてみましょう。オレンジは赤と黄色を1ドットごとに交互に塗ればよさそうです。

オレンジのパターン

赤	黄	赤	黄	赤	黄	赤	黄

赤と黄は、例の表を使って原色と輝度に分解すると次のようになります。

	赤	黄	赤	黄	赤	黄	赤	黄	16進数読み
B	0	0	0	0	0	0	0	0	→ &H00
G	0	1	0	1	0	1	0	1	→ &H55
R	1	1	1	1	1	1	1	1	→ &HFF
I	0	0	0	0	0	0	0	0	→ &H00

この表を横に16進数で読んで文字に変換し加えたものが、赤と黄色の1ドット模様(人間の目にはオレンジに見える)を表すタイルパターンになります。

タイルパターン(オレンジ)

```
TILE$ = CHR$(&H0F) + CHR$(&H55) + CHR$(&HFF) + CHR$(&H00)
```

【例題121】　中間色塗り　タイルパターン

画面中央に半径100の緑の円を描き、内部をオレンジに塗ってください。

解答例

```
CLS
CIRCLE (320, 200), 100, 4
TILE$ = CHR$(&H00) + CHR$(&H55) + CHR$(&HFF) + CHR$(&H00)
PAINT (320, 200), TILE$, 4
```

解説　■よーく見ると

実行させてできたオレンジの丸を目を近づけてよーく見てください。縦に縞模様が見えませんか。これは、赤と黄色が縦方向に揃っているからです。

できれば縦方向にも、赤と黄色を交互にすればもっと滑らかなオレンジになります。次のセクションでは、タイルパターンを縦方向にも設定することを学習します。

タイルパターン(2)

ペイント文によるタイルパターンの指定は横方向には8ドット単位ですが、縦方向には64ドットまで指定できます。赤と白の市松模様を考えてみましょう。

市松模様

赤	赤	赤	赤	白	白	白	白
赤	赤	赤	赤	白	白	白	白
赤	赤	赤	赤	白	白	白	白
赤	赤	赤	赤	白	白	白	白
白	白	白	白	赤	赤	赤	赤
白	白	白	白	赤	赤	赤	赤
白	白	白	白	赤	赤	赤	赤
白	白	白	白	赤	赤	赤	赤

横に分解　上のパターンは赤白のパターンと白赤のパターンに分解できます。

(1)赤白のパターン

	赤	赤	赤	赤	白	白	白	白	
B	0	0	0	0	1	1	1	1	&H0F
G	0	0	0	0	1	1	1	1	&H0F
R	1	1	1	1	1	1	1	1	&HFF
I	0	0	0	0	0	0	0	0	&H00

T1$ = CHR$(&H0F) + CHR$(&H0F) + CHR$(&HFF) + CHR$(&H00)

(2)白赤のパターン

	白	白	白	白	赤	赤	赤	赤	
B	1	1	1	1	0	0	0	0	&HF0
G	1	1	1	1	0	0	0	0	&HF0
R	1	1	1	1	1	1	1	1	&HFF
I	0	0	0	0	0	0	0	0	&H00

T2$ = CHR$(&HF0) + CHR$(&HF0) + CHR$(&HFF) + CHR$(&H00)

縦に合成　この各パターンを上から加えたものが、求める赤白の市松模様のタイルパターンになります。

タイルパターン(赤白の市松模様)

```
T1$ + T1$ + T1$ + T1$ + T2$ + T2$ + T2$ + T2$
```

【例題122】 模様塗り　タイルパターン

(320, 200) の半径100の円内部を、左ページの(例)のタイルパターン(赤白の市松模様)で塗ってください。

解答例

```
CLS
CIRCLE (320, 200), 100, 4
T1$ = CHR$(&H0F) + CHR$(&H0F) + CHR$(&HFF) + CHR$(&H00)
T2$ = CHR$(&HF0) + CHR$(&HF0) + CHR$(&HFF) + CHR$(&H00)
TILE$ = T1$ + T1$ + T1$ + T1$ + T2$ + T2$ + T2$ + T2$
PAINT (320, 200), TILE$, 4
```

【例題123】 中間色塗り　タイルパターン

(320, 200)に半径100の緑の円を描き、内部をオレンジに塗ってください。

オレンジのパターン（【例題121】よりなめらかなオレンジ）

赤	黄	赤	黄	赤	黄	赤	黄
黄	赤	黄	赤	黄	赤	黄	赤

(1) 赤黄のパターン

	赤	黄	赤	黄	赤	黄	赤	黄	
B	0	0	0	0	0	0	0	0	&H00
G	0	1	0	1	0	1	0	1	&H55
R	1	1	1	1	1	1	1	1	&HFF
I	0	0	0	0	0	0	0	0	&H00

(2) 黄赤のパターン

	黄	赤	黄	赤	黄	赤	黄	赤	
B	0	0	0	0	0	0	0	0	&H00
G	1	0	1	0	1	0	1	0	&HAA
R	1	1	1	1	1	1	1	1	&HFF
I	0	0	0	0	0	0	0	0	&H00

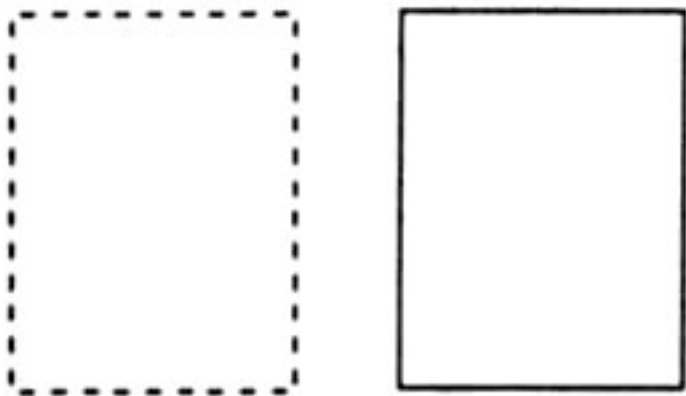
```
CLS
CIRCLE (320, 200), 100, 4
TILE1$ = CHR$(&H00) + CHR$(&H55) + CHR$(&HFF) + CHR$(&H00)
TILE2$ = CHR$(&H00) + CHR$(&HAA) + CHR$(&HFF) + CHR$(&H00)
TILE$ = TILE1$ + TILE2$
PAINT (320, 200), TILE$, 4
```

別解　＆や＄などをたくさん打つのが面倒な人へ

```
CIRCLE (320, 200), 100, 4
TILE$ = ""
FOR I = 1 TO 8
    READ P$
    TILE$ = TILE$ + CHR$(VAL("&H"+P$))
NEXT
PAINT (320, 200), TILE$, 4
DATA 0, 55, FF, 0, 0, AA, FF, 0
```

解説　**■滑らかなオレンジ**

【例題121】より色塗りが滑らかになっているとは思いませんか。ふたつ並べて比べてみてください。

アニメーション

それでは、本書の最後のテーマ、アニメーションの基礎に移ります。

アニメーションすなわち動画は、次のように絵を場所を変えて再描画することによって実現できます。

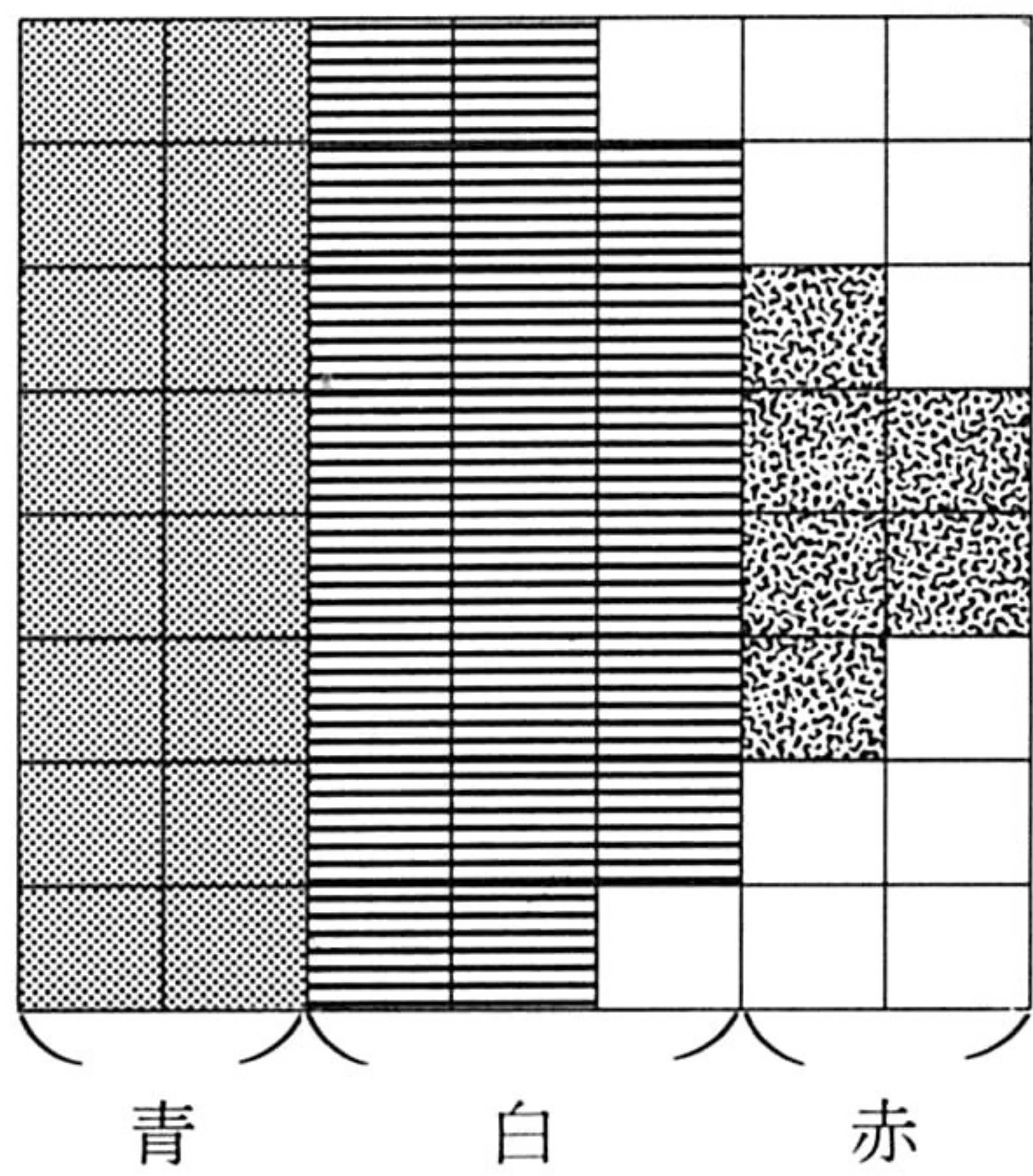
(1) 描く

(2) 消して、となりに描く

(3) 消して、となりに描く

円とか四角ならCIRCLE文やLINE文による描画を上の手法で動かすことができますが、絵ともなると単純なCIRCLEやLINEでは表現できません。ドット単位で描いた絵をGET(取り上げて)して、別の位置へPUT(描き出す)することになります。このセクションでは、Quick BASIC最後の学習として、ピストルの弾を左から右に動かしてみましょう。

ピストルの弾

青　白　赤

ピストルの弾は、左図のように青、白、赤からなるこんな形のものとします。

原画の作図

まず、このピストルの弾を画面のどこか(ここでは(0, 0)を左上とする位置)に描きます。

ピストルの弾のパターンと色

(0,0) (7,0)

0	青	青	白	白			
0	青	青	白	白	白		
0	青	青	白	白	白	赤	
0	青	青	白	白	白	赤	赤
0	青	青	白	白	白	赤	赤
0	青	青	白	白	白	赤	
0	青	青	白	白	白		
0	青	青	白	白			

(0,7) (7,7)

同左カラーコード表示

(0,0) (7,0)

0	1	1	7	7			
0	1	1	7	7	7		
0	1	1	7	7	7	4	
0	1	1	7	7	7	4	4
0	1	1	7	7	7	4	4
0	1	1	7	7	7	4	
0	1	1	7	7	7		
0	1	1	7	7			

(0,7) (7,7)

【例題124】 アニメーション 原画の作図

ピストルの弾を、画面の左上に描いてください。

解答例

```
DATA  0, 1, 1, 7, 7, 0, 0, 0
DATA  0, 1, 1, 7, 7, 7, 0, 0
DATA  0, 1, 1, 7, 7, 7, 4, 0
DATA  0, 1, 1, 7, 7, 7, 4, 4
DATA  0, 1, 1, 7, 7, 7, 4, 4
DATA  0, 1, 1, 7, 7, 7, 4, 0
DATA  0, 1, 1, 7, 7, 7, 0, 0
DATA  0, 1, 1, 7, 7, 0, 0, 0
FOR I = 0 TO 7
      FOR J = 0 TO 7
            READ C
            PSET (J, I), C
      NEXT J
NEXT I
```

配列への格納　GET

配列

このピストルの弾を動かすためには、このピストルの弾の絵自体をパターンとして配列に格納することから始めます。

配列の大きさ

絵をしまうのにどれだけの配列が必要かは、次の式で計算します。

> 配列の大きさ＝{4+4＊((幅＋7)/8)＊高さ}/2
>
> ↑ ((幅＋7)/8) の小数部を切り捨てる

ピストルの絵は横８、縦８の大きさですから、必要な配列の大きさは18になります。

配列の大きさ＝{4＋4＊((8＋7)/8＊8)}/2
＝{4＋4＊8}/2
＝(4＋32)/2
＝18

配列への格納　GET

配列が決まると、その配列に絵のイメージを取り込む命令 GET が使えます。

イメージの取り込み

> GET（パターン左上）－（パターン右下），配列名

【例題125】　絵のイメージを取り込む　GET

> 画面左上に描かれたピストルの弾を、配列に格納してください。そして左上のピストルの弾を消してください。

解答例

先のピストルの弾を描くプログラムに続けて、

```
DIM  Image(18)
GET (0, 0) - (7, 7), Image
CLS
```

配列の描き出し　PUT

配列にしまった絵のイメージは、PUT によって任意の場所に描き出すことができます。

配列の描き出し

```
PUT (座標), 配列, XOR
```

最後に付いている XOR(エックスクルージブオア)は、絵の背景を残すための手段です。

【例題126】　絵のイメージを出力する　PUT

配列にしまってあるピストルの弾を、画面中央の左から右へひとつおきにつなげてください。

解答例

ここまでのプログラムに続けて、

```
FOR I = 0 TO 639 STEP 16
    PUT (I, 200), Image, XOR
NEXT I
```

再描画による消し込み

同じパターンを同じ位置に再描画すると、その絵から消えてなくなります。正確には、背景が再現されます。これは、PUT に付いている XOR の効果ですから、 必ず XOR を付けて PUT 文を使用してください。

【例題127】 再描画による消し込み

ピストルの弾を、画面中央で左から右に動かしてください。

解答例

ここまでのプログラムの最後の FOR 文を、次のように修正してください。

修正前

```
FOR I = 0 TO 639 STEP 16
      PUT (I, 200), Image, XOR
NEXT I
```

修正後

```
FOR I = 0 TO 623 STEP 16
      FOR K = 1 TO 10
      NEXT
      PUT (I+16, 200), Image, XOR
      IF I <> 0 THEN
          PUT (I, 200), Image, XOR
      END IF
NEXT I
```

索　引

記号・アルファベット順

五十音順

あ行

か行

ま行

や行

ら行

著者紹介

小島政行(こじま　まさゆき)

1949年生まれ。慶応義塾大学工学部計測工学科卒。
1988年アプライド ナレッジ社設立、代表取締役に就任。
エキスパート構築システム、リレーショナル・データベースなどのマニュアル、解説書の作成などに活躍中。

アクセス　ブックス

Quick　BASIC プログラミングテキスト

1989年7月26日　初版発行
1993年5月9日　第2刷

著　者　小島政行
発行者　川田　良
発行所　サイエンテック
〒142　東京都品川区西中延2－2－16
振　替　東京9－110881
電　話　03－3782－3361

印　刷　株式会社耕文社

サイエンテック アクセスブックス

dBASE Ⅲ PLUS システム手帳

中島正明著　判型Ｂ５　定価 2,060円(本体2,000円) (別売ディスクあり)

システム手帳用アプリケーション(カレンダー・スケジュール管理・日記帳(日誌)・電話帳)とプログラム管理ユーティリティを掲載。

dBASE Ⅲ PLUS デベロッパーズ ツールキット

葛西秋雄著　判型Ｂ５　定価 2,900円 (本体2,816円) (別売ディスクあり)

アプリケーションを開発するためのプログラミング・ツールキットを掲載。

dBASE Ⅲ PLUS 情報整理学〈テキストデータベース〉

中島正明著　判型Ｂ５　定価 2,500円 (別売ディスクあり)

文字（テキスト）データを有効に生かすためのアプリケーションを掲載。

dBASE Ⅲ PLUS 応用テクニックブック

酒井　良著　判型Ｂ５変形　定価 2,500円 (別売ディスクあり)

プログラミングのための便利なライブラリやプログラム・ジェネレータを紹介。

dBASE Ⅲ PLUS 活用ガイドブック

酒井　良著　判型Ｂ５変形　定価 2,400円 (本体2,330円)

dBASE III PLUS をわかりやすく解説した入門書。

dBASE Ⅲ PLUS コンパイラ＆アセンブラ

桑村幸雄・池端良一著　判型Ｂ５　定価 2,900円 (本体2,816円) (別売ディスクあり)

dBASE III PLUS のコンパイル手法とアセンブラの応用を解説。

dBASE Ⅲ 基本ワザ全集〈顧客編〉

桑村幸雄著　判型Ｂ５変形　定価 2,300円 (別売ディスクあり)

顧客管理を作成しながら、プログラミングをマスター。顧客管理プログラム全掲載。

dBASE Ⅲ 裏ワザ全集

酒井　良著　判型Ｂ５変形　定価 2,000円 (別売ディスクあり)

dBASE IIIをフル活用するための応用テクニック・ショートプログラムを掲載。

Quick Cが不思議と良くわかる本

河野春夫著　判型Ｂ５　定価 2,060円（本体2,000円）

Quick Cの基本操作から住所録管理プログラム作成まで掲載した入門書。

EGR98ライティッシュガイド

河野春夫著　判型Ａ５　定価 1,600円

グラフィック・ツールとしてのEGR98を手軽に使うための入門書。付録として EGR98をdBASE III PLUSで使用するためのインターフェイスプログラムを掲載。

入門FRAME WORK II

兼子紀美著　判型Ａ５　定価 2,000円

はじめてFRAME WORK IIを使う人のための入門書。初心者のために環境設定からやさしく解説。

Z'S STAFF Kidハンドブック

沖田新一著　判型Ｂ５変形　定価 2,000円

基本操作から実務での応用例、技術情報まで詳細に解説した実用書。

花子 基本ワザ全集

北大パソコン研究会著　判型Ｂ５変形　定価 2,300円

基本的な知識から応用例・事例集まで満載、花子を完全にマスターできる本。

■全国有名書店、マイコンショップにてお求め下さい。

■直接当社へお申し込みの場合は、下記の郵便振替口座にお振り込み下さい。

送料に関しては、何冊お求めになられても250円です。

〒142　品川区西中延２－２－16

TEL 03―782―3361　郵便振替　東京９－110881

＊本広告の中で本体価格が記載されていない書籍は、消費税抜きです。
お買上の際は、消費税込み定価に改定している場合がありますが、ご了承下さい。

サイエンテック ソフトウェア

dBASE Ⅲ PLUS システム手帳

PC-9800 5″/3.5″ 2HD 定価10,000円

「dBASE III PLUS システム手帳」に掲載プログラム〈カレンダープログラム・スケジュール管理プログラム・日記帳（日誌）プログラム・電話帳プログラム・プログラム管理プログラム〉の計5本のプログラムを収録。

dBASE Ⅲ PLUS デベロッパーズ・ツールキット

PC-9800 5″／3.5″ 2HD 定価15,000円

「dBASE III PLUS DEVELOPER'S TOOL kit」に掲載プログラムとサンプルプログラム1とサンプルプログラム2を収録。サンプルプログラム1は、ツールキットを利用して作成した簡単な例。サンプルプログラム2は、ツールキットを利用してグラフィックデータ管理までできる完成したアプリケーション(図書管理システム)です。

dBASE Ⅲ PLUS テキスト・データベース プログラム・ディスク

PC-9800 5″／3.5″ 2HD 定価12,000円

「dBASE III PLUS 情報整理学」に掲載のテキスト・データベースプログラムをフロッピー版・ハードディスク版・疑似コンパイル版に分けて収録。

dBASE Ⅲ PLUS 応用テクニックブック プログラム・ディスク

PC-9800 5″ HD 定価12,000円

「dBASE III PLUS 応用テクニックブック」に掲載のライブラリ集を収録。

dBASE Ⅲ V2.1J・PLUS 顧客管理システム

PC-9800 5″ 2HD 定価12,000円

「dBASE III基本ワザ全集〈顧客編〉」に掲載の顧客管理プログラムを収録。

dBASE Ⅲ 顧客管理システム コンパイル版

PC-9800 5″ 2HD 定価24,000円

上記顧客管理システムのクイックシルバーによるコンパイル版。ソースプログラムとコンパイル済実行ファイルを収録。

dBASE Ⅲ PLUS コンパイラ&アセンブラ プログラム・ディスク

PC-9800　5″ 2HD　3枚組　定価15,000円

「dBASE III PLUS コンパイラ&アセンブラ」に掲載のプログラムを収録。

dBASE Ⅲ 裏ワザ全集 プログラム・ディスク

PC-9800　5″ 2HD　定価 7,800円

「dBASE III裏ワザ全集」に掲載のプログラムを収録。

R:BASE5000 顧客管理システム

PC-9800　5″ 2HD　定価12,000円

「R：BASE5000基本ワザ全集〈顧客編〉」に掲載の顧客管理プログラムを収録。

■各ソフトウェアをお求めになる場合、本誌に添付されている郵便振替用紙をご使用下さい。なお、用紙がない場合は、郵便局の備え付けの振替用紙にて下記の郵便振替口座にお振り込み下さい。

口座名　　株式会社サイエンテック

郵便振替　東京９―110881

＊ソフトウェアに関しては、消費税及び送料はいただいておりません。

＊商品をお急ぎの方は、振替用紙の領収書と商品名・メディアサイズ・住所・氏名をハッキリと書いて下記の番号にFAXして下さい。確認しだい商品を発送いたします。

FAX番号　　03―782―4109

例題方式

Quick BASIC プログラミングテキスト

小島政行著

ISBN4-88244-025-3 C3055 P2060E

定価2,060円(本体2,000円)